MF3E(H)x3

MIFARE DUOX contactless smartcard IC

Rev. 1.2 — 10 July 2025
974412

Product data sheet
CONFIDENTIAL

Protected by NDA only
CONFIDENTIAL - Personal copy for:
Sergio Sala
Quantum Solutions GmbH
Do not copy, share or reproduce
5fc077a2-2c4b-90e-6ee3cae4f1

1 General description

1.1 Introduction

MF3E(H)x3 is the latest addition to the MIFARE product family introducing new features and new crypto protocols by offering ECC asymmetric cryptography and PKI support, along with enhanced performance for best user experience. As the first product in the MIFARE family supporting asymmetric cryptographic features, MIFARE DUOX opens up new use cases and allows simplified key management as well as simplified key distribution approaches, relying on PKI functionality.

MF3E(H)x3 supports both symmetric as well as asymmetric cryptography in form of AES128, AES256 or ECC cryptographic protocols. For asymmetric cryptographic support, the IC features keypair generation, signature generation and verification as well as PKI based X.509 certificate handling. The newly introduced asymmetric cryptographic protocols allow for a simplified key management and increased system security as the risk associated with secret key leakage from the reader terminal that can be abused for credential cloning is eliminated.

The MF3E(H)x3 is Common Criteria EAL6+ security certified which is the same security certification level as demanded for smart card IC products used, e.g., for banking cards or electronic passports. It fully complies with the requirements for fast and highly secure data transmission and flexible application management. This makes it the ideal product for service providers and service operators who want to offer an easy, convenient, and secure access to a wide variety of different services, but still relying on a high secure product, without compromising on security.

MF3E(H)x3 offers best flexibility when creating multi-application schemes on one smart card IC. Features such as Delegated Application Management are supporting flexible business models. Smart City services could be utilized with only one smart card by combining services such as access applications (access to city attractions, corporate and governmental access), car or bike sharing, citizen services, car access applications and electric vehicle charging applications.

MF3E(H)x3 is based on global open standards for both air interface and cryptographic methods. It is compliant to all levels of ISO/IEC 14443A and supports optional ISO/IEC 7816-4 commands (APDU and file structure supported) and is fully interoperable with existing NFC reader installations for MIFARE infrastructure. Additionally, MF3E(H)x3 is compliant with NFC Forum Tag Type 4 and obtained the NFC Forum certification, as seen in Figure 1, allowing creation of NDEF formatted applications on the IC.

Depending on the chosen delivery and package type, MF3E(H)x3 features can be utilized via the NFC interface or via the newly introduced I^2C interface (which is optionally available on selected product configurations).

Featuring an on-chip backup management system and the mutual authentication, a MF3E(H)x3 card can hold as many applications as the memory can accommodate. Each application can hold up to 32 files with various file configurations and different settings that can be applied. The size of each file is defined at the moment of its creation, making MF3E(H)x3 a truly flexible and convenient product. An automatic anti-tearing mechanism is available for all file types, guaranteeing transaction-oriented data integrity.

The main characteristics of this device are underpinned by the outstanding position of the MIFARE product portfolio as fast, innovative, reliable, and secure smart card ICs in the contactless proximity transaction market.

MF3E(H)x3 delivers the perfect balance of speed, performance, and cost efficiency. Its open concept allows seamless future integration of other smart card media such as smart paper tickets, banking convergence card, and MIFARE 2GO mobile ticketing service based on Near Field Communication (NFC) technology. MF3E(H)x3 is your entrance to secure contactless systems worldwide.

Figure 1. NFC Forum certification for MF3E(H)x3

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

2 Features and benefits

2.1 Feature overview

2.1.1 RF interface: ISO/IEC 14443 Type A

- Contactless interface compliant with ISO/IEC 14443-2/3/4 A
- Contactless transmission protocol using ISO/IEC 14443-4
- Low Hmin enabling operating distance up to 100 mm (depending on power provided by the PCD and antenna geometry)
- Fast data transfer: 106 kbit/s, 212 kbit/s, 424 kbit/s, 848 kbit/s
- Very High Bit Rate (VHBR): 1.7 Mbit/s and 3.4 Mbit/s - PICC to PCD only
- Configurable FSCI to support up to 256 bytes frame size

2.1.2 Non-volatile memory

- 2 kB, 4 kB or 8 kB standard memory sizes are available
- 16 kB memory size is available for applications requesting a large user memory size
- Data retention of 25 years
- Write endurance of typical 1 000 000 cycles

2.1.3 NV-memory organization and multi-application support

- **Flexible application management on the IC**, see Section 7.11
  - A flexible file system offering the user free definition of application and file structures on the PICC
  - The number of applications is not limited: applications can be created as long as there is free user memory available on the PICC
  - Shared application management allows to access files from any two applications during a single transaction if access rights are configured accordingly, see Section 7.11.3.1
- **Flexible and dynamic file management inside the applications**, see Section 7.12
  - The amount of files per application is set to 32: up to 32 files can be created in each application
  - Availability of 6 file types: Standard data file, Backup data file, Value file, Linear Record file, Cyclic Record file and Transaction MAC file
  - File size is determined during the file creation (exception for Transaction MAC file)
- **Delegated Application Management feature** allows smart management of multiple applications per smart card shared by different entities, see Section 7.11.2
  - Memory can be re-used in delegated applications (via formatting the complete delegated application)
  - Factory loaded NXP's Delegated Application Management (DAM) keys for a remote application management service support

2.1.4 Security and privacy

- Common Criteria certification: CC EAL6+ AVA_VAN.5 on Hardware and Software
- 7 bytes unique identifier (UID) for each device
- Optional 4 bytes random ID can be enabled, for enhanced security and privacy of the device
- Hardware exception sensors are in place
- Self-securing file system is implemented
- **Symmetric cryptography features**, see Section 7.8:
  - Mutual AES-based three-pass authentication

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Serial: Sandra Quandt
Do not copy, share or reproduce
Quandt IT Solutions GmbH
5fc077ad-c22c-4056-50fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

- Flexible symmetric key management: 1 card master key and up to 14 application keys per application key set
- Multiple key sets per application with fast key rolling mechanism (up to 16 key sets per application)
- Hardware AES co-processor supporting AES 128-bit keys or AES 256-bit keys

• **Asymmetric cryptography features**, see Section 7.9:
- Mutual and Reader-Unilateral ECC-based authentication
- Card-Unilateral ECC-based authentication
- Flexible certificate / asymmetric keypair management and certificate management, see Section 7.10:
  - Up to two CA Public Keys at PICC level
  - Up to five CA Public Keys per application
  - Up to five private keys per application
- ECC key pair generation over NIST curve P-256 and brainpoolP256r1
- Hardware ECC co-processor supporting ECC 256-bit keys
- ECC-based Originality Check feature ensures proof of genuine NXP products

• **Access Rights Management**, see Section 7.6:
- Multiple key assignments for each file access rights (up to 8)
- Unified access right management system allowing for mixed applications with symmetric- and asymmetric-based authentication access control

• **Secure Messaging**, see Section 7.5:
- Secure channel with AES-based data encryption and data authenticity ensured by CMAC
- Authentication can be executed on card and on application level

• **Security-related features:**
- Transaction MAC and Transaction Signature features are available for symmetric and asymmetric cases, ensuring generating secure checksum / secure signature per application and executed transaction, see Section 7.14.3 and Section 7.14.4
- Transaction Timer feature is available to protect against Man-in-the-Middle attacks, see Section 7.14.5
- Proximity Check feature is offered for protection against Relay Attacks, see Section 7.18

• **Originality Check feature ensuring authenticity of the product**, see Section 7.21
- Dynamic ECC-based Originality Check to ensure proof of genuine NXP products
- MIFARE DUOX is pre-provisioned with an Originality Check ECC-based key pair and related certificate, allowing the verification of the genuineness of the IC
- Executed via a card-unilateral authentication through a challenge-response protocol

2.1.5 ISO/IEC 7816 compatibility

• Support for standardized ISO/IEC 7816-4 standard as explained in Section 7.17
- Supports ISO/IEC 7816-4 file structure (selection by File ID or by DF name)
- Supports ISO/IEC 7816-4 APDU message structure
- Supports ISO/IEC 7816-4 APDU wrapper for MF3E(H)x3 native commands
- Supports ISO/IEC 7816-4 INS code ‘A4’ for SELECT FILE
- Supports ISO/IEC 7816-4 INS code ‘B0’ for READ BINARY
- Supports ISO/IEC 7816-4 INS code ‘D6’ for UPDATE BINARY
- Supports ISO/IEC 7816-4 INS code ‘B2’ for READ RECORDS
- Supports ISO/IEC 7816-4 INS code ‘E2’ for APPEND RECORD
- Supports ISO/IEC 7816-4 INS code ‘84’ for GET CHALLENGE
- Supports ISO/IEC 7816-4 INS code ‘87’ for GENERAL AUTHENTICATE
- Supports ISO/IEC 7816-4 INS code ‘88’ for INTERNAL AUTHENTICATE

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Joerg Sauer
Giesecke+Devrient Solutions GmbH
Charge-on share or reproduce
Document id: 5fc07b0c-22c4-405c-9ffe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

2.1.6 Special features

MF3E(H)x3 offers a wide variety of available features that can be consumed and configured individually on the IC. The flexibiliy of feature-availability allows a customization of the product according to the needs and requirements of the customer.

- **Various configuration options on the IC via the SetConfiguration command**, see Section 7.7:
  - Configurable ATS and contactless protocol parameter (ATQA, SAK) configuration: Offering flexible NFC parameter settings to fit the customer`s NFC reader infrastructure requirements
  - Secure Messaging related configuration: Offering selection of authentication and secure messaging related detail setting, including curve selection for asymmetric authentication protocol, optional asymmetric private key usage limitation, and authentication counter setting
  - Card and application level configuration: Offering specific settings to apply configurations for ISODFName, VCIID, Capability Data, Application Default Keys, etc
  - Communication channel configuration: Offering settings to define NFC, I^2C and GPIO related detailed settings (if applicable to the product type / product configuration, and if required)
- **Inclusion of asymmetric cryptography to the product**, see Section 7.9 and Section 7.10:
  - Aditionally to the symmetric AES128 and AES256 functionality also ECC asymmetric cryptography is supported
  - Major functionality related to ECC is added to MF3E(H)x3, listed in detail in Section 7.1 (e.g. ECC-based authentication, certificate management, asymmetric key management, PKI functionality etc)
- **Transaction Signature feature**, see Section 7.14.4:
  - Similar functionality offered as via the symmetric Transaction MAC feature which is consuming AES128 or AES256 symmetric crypto. Transaction Signature is based on asymmetric ECDSA signature which is calculated on top the executed transaction, allowing for reduced security requirements on the reader / back-end infrastructure storage, as only a public key is required for validation
- **Transaction Timer feature**, see Section 7.14.5:
  - Feature to prevent Man-in-the-Middle (MitM) attacks where the attacker delays the conclusion of a transaction by keeping the card powered after it left the legitimate reader device
- **Secure Unique NFC (SUN) enabled by Secure Dynamic Messaging (SDM)** to ensure confidential and integrity-protected data exchange without prior authentication, see Section 7.5.11. There are two flavors of the feature available on MF3E(H)x3:
  - Symmetric SDM MAC: data is protected by a MAC generated by a symmetric AES key
  - Asymmetric SDM SIG: data is protected by a ECDSA signature generated by the private key of an ECC key pair
  - SUN / SDM allows adding security to the data that´s read out from the chip, while still being able to access it with standard NDEF readers for NFC Forum Tag Type 4 formatted cards.
  - The typical use case is an NDEF holding a URI and some meta-data, where SDM allows this meta-data to be communicated confidentiality and integrity protected toward a backend server.
  - The SDM feature also allows the realization of **Second Factor Authentication (2FA)** and **One-Time-Password (OTP)** schemes. In these scenarios, MIFARE DUOX IC can be used as secure medium to generate a 2FA code or OTP for secure user login to a device or service (**Logical Access**).
- **NFC Forum Type 4 Tag certification of the IC**, allowing to store NDEF Tag Type 4 formatted messages on the IC, for an automated interaction when tapping the smartcard to an NFC enabled mobile device
- **Product versions** offering both the low input and high input capacitance versions (17pF and 70pF):
  - Allowing to optimally tune smartcard IC antennas for any kind of form factor size and design (standard ID-1 sized smartcards or smaller form factors like wristbands, key fobs, tokens, etc)
- **Compatibility to MIFARE DESFire products:**
  - MF3E(H)x3 is based on the same underlying file system concept as existing products in the MIFARE DESFire product family. The concept of applications and files that can be created on the PICC is backwards-compatible.

Provided under NDA only - COMPANY CONFIDENTIAL - Personal copy for: Setel Data IT Solutions GmbH - Do not copy, share or forward - 5fc77bd-c2c-4056-9bfd-6ec30ea2ff

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07790-c22c-405c-90fe-6ecc3cae42f1

- MF3E(H)x3 is functional compatible to the AES-128 cryptographic mechanisms of the MIFARE DESFire EV3 product. Support and compatibility for DES, 2K3DES and 3K3DES cryptographic mechanisms has been disabled on MF3E(H)x3.
- Existing card layouts that are based on MIFARE DESFire EV3 utilizing AES-128 cryptography can be ported to MIFARE DUOX.

- **Compliancy to Automotive Industry regulations and requirements:**
  - Compliancy of MF3E(H)x3 to ISO / SAE 21434 cyber security regulations
  - Compliancy to the MISRA-C coding standard
  - Dedicated Automotive-like qualification on silicon level of MF3E(H)x3, which is based on a dedicated qualification flow including cold test
  - Support of an extended temperature range from -40°C up to 105°C operating condition on silicon level, which is the perfect fit for automotive environments
  - Fast ECC-based authentication for initial pairing between smartcard IC and vehicle
  - Proximity Check feature as key functionality to prevent Relay Attacks in the automotive segment
- **Compliancy to Electric Vehicle Charging (EV-Charging) Industry regulations and requirements**, see Section 7.19:
  - Utilizing PKI capability of MF3E(H)x3 for enabling interoperability between multiple CPOs and EMSPs
  - Seamless feasibility to use one smartcard in multiple EV-Charging networks (tremendous end user benefit)
  - Compliancy to the VDE-DKE regulation VDE-AR-E 2532-100 (support of required file structure and command support for electric vehicle charging applications)
    - Availability of the required command set to interact with EV-Charging stations as per the mentioned guideline
    - Provisioning of required card layout, application structure, configuration settings, asymmetric keypair and certificate during the NXP Trust Provisioning process (available in a dedicated product type with dedicated part number)

2.1.7 I²C interface

In specific product types / product configurations of MF3E(H)x3, an optional I²C interface is available and can be used.

Please refer to Section 2.2, for the availablility of the I²C interface in different package types.

If the I²C interface, refer to detailed description in Section 7.4, is available in the product configuration, following points apply:

- MF3E(H)x3 optionally acts as I²C follower using a 7-bit address
- I²C interface operates at 100 kHz (standard mode), 400 kHz (fast mode) or 1 MHz (fast mode plus) and consists of two signals SDA (data) and SCL (clock)

2.1.8 GPIO

In specific product types / product configurations of MF3E(H)x3, GPIOs are available and can be used.

Please refer to Section 2.2, for the availablility of the GPIOs in different package types.

If the GPIOs, refer to detailed description in Section 7.16, are available in the product configuration, following points apply:

- Two configurable GPIOs (GPIO1 and GPIO2) are available via OTP
- Configuration options: input/output direction, debounce, output type, drive strength and pull-up/down configuration
- One GPIO can be used for power downstream with up to 10 mW and can be enabled via NFC command

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

2.2 Feature to package mapping

MF3E(H)x3 is a platform product which is made available in different packages, offering a variety of different features.

An overview of the MF3E(H)x3 packages mapped to available features and functionality offered by the package is available in Table 1.

Table 1. MF3E(H)x3 available packages and features

Features	75 µm bumped wafer	MOA8 contactless module	WLCSP package	HVQFN20 package
NFC interface	x	x	x	x
I^2C interface			x	x
GPIO1	x		x	x
GPIO2			x	x
Power downstream	x		x	x

All packages and delivery types are described in detail in [36].

At this time, the following delivery types are released and available for ordering, as indicated in Section 4:

- **75µm bumped wafer** (12 inch wafer; sawn; bumped; 75 µm thickness)
- **MOA8 contactless module** (plastic leadless module carrier package of type MOA8)

The additional packages of type **WLCSP** and **HVQFN20** can be made available upon request.

Provided under NDA only - COMPANY CONFIDENTIAL - Personal copy for: Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

3 Target applications

MF3E(H)x3 is the future proven secure contactless smart card IC for applications demanding flexibility, dynamic key provisioning and high security.

The potential application areas and use case possibilities vary from a wide spectrum of available scenarios. Depending on client and customer requirements, MF3E(H)x3 can be positioned in countless different applications as the product' s features and configuration options give the broadest possibilities to tailor the product to the individual end customer requirements.

Ideal and exemplary target application are

- Physical Access Management (corporate, governmental, high-security areas, etc.)
  - Especially aiming at multi-site Access Management systems, MF3E(H)x3 is the perfect fit to realize versatile and enhanced access infrastructures.
- Secure Car Access Management and Digital Car Key Management (secure access to vehicle and start of engine)
- Electric Vehicle Charging (user authorization to start EV-charging session, EV-charging credential, EV-charging membership card, etc.). For the EV-Charging vertical, a dedicated product configuration is available, which features pre-loaded data and keys inside the MIFARE DUOX user memory. Details on this can be found in Section 7.19.4.
- Car and Vehicle Sharing (car, scooter, electric vehicle sharing applications, etc)
- Public Transportation
- Micropayment
- Loyalty Schemes
- Smart City Applications
- Security Applications

Provided under NDA only / COMPANY CONFIDENTIAL - Personal copy for: Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-405c-90fe-6ecc3ca0211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

4 Ordering information

MF3E(H)x3 is available in one major product version which can be used for all use cases and verticals.

An additional, second trust-provisioned configuration is available, which is building on top of the major product version. This product version includes trust-provisioned (pre-loaded) EV-Charging specific data structure, keys and related certificates.

The product configuration information is part of the product version information which can be read-out from the IC with the GetVersion command, as defined in Section 7.7.2.1. Further details can be found in Section 7.7.2. Retrieving all details of the product version is possible with the command GetVersion as defined in Section 8.3.5.

The MIFARE DUOX MF3E(H)x3 product is characterized through following aspects:

- Extended product qualification flow meeting automotive industry requirements
- Additional cold-test during product qualification
- Extended temperature range for operating conditions of IC: -40°C up to +105°C on silicon level

The product type list and ordering information is available in following subsections:

- MIFARE DUOX ordering information: Section 4.1
- MIFARE DUOX for EV-Charging ordering information: Section 4.2

4.1 Ordering information for MIFARE DUOX MF3E(H)x3 product

Table 2. Ordering information for MIFARE DUOX

Type number	Package		
	Name	Description	Version
MF3E23A1DUF/01	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 2 kB, 17 pF input capacitance	-
MF3E43A1DUF/01	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 4 kB, 17 pF input capacitance	-
MF3E83A1DUF/01	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 8 kB, 17 pF input capacitance	-
MF3E93A1DUF/01	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 16 kB, 17 pF input capacitance	-
MF3EH23A1DUF/01	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 2 kB, 70 pF input capacitance	-
MF3EH43A1DUF/01	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 4 kB, 70 pF input capacitance	-
MF3EH83A1DUF/01	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 8 kB, 70 pF input capacitance	-
MF3EH93A1DUF/01	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 16 kB, 70 pF input capacitance	-
MF3E23A0DA8/01	MOA8	plastic leadless module carrier package[3]; 2 kB, 17 pF input capacitance	SOT500-4
MF3E43A0DA8/01	MOA8	plastic leadless module carrier package[3]; 4 kB, 17 pF input capacitance	SOT500-4

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07799-2ac4-4b8e-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 2. Ordering information for MIFARE DUOX*...continued*

Type number	Package		
	Name	Description	Version
MF3E83A0DA8/01	MOA8	plastic leadless module carrier package[3]; 8 kB, 17 pF input capacitance	SOT500-4
MF3E93A0DA8/01	MOA8	plastic leadless module carrier package[3]; 16 kB, 17 pF input capacitance	SOT500-4
MF3EH23A0DA8/01	MOA8	plastic leadless module carrier package[3]; 2 kB, 70 pF input capacitance	SOT500-4
MF3EH43A0DA8/01	MOA8	plastic leadless module carrier package[3]; 4 kB, 70 pF input capacitance	SOT500-4
MF3EH83A0DA8/01	MOA8	plastic leadless module carrier package[3]; 8 kB, 70 pF input capacitance	SOT500-4
MF3EH93A0DA8/01	MOA8	plastic leadless module carrier package[3]; 16 kB, 70 pF input capacitance	SOT500-4

[1] Delivered on film frame carrier with electronic fail die marking according to SECSII format.
[2] See [36]
[3] See Figure 99

4.2 Ordering information for MIFARE DUOX EV-Charging MF3E(H)x3../01EV product

Table 3. Ordering information for MIFARE DUOX for EV-Charging

Type number	Package		
	Name	Description	Version
MF3E23A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 2 kB, 17 pF input capacitance	-
MF3E43A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 4 kB, 17 pF input capacitance	-
MF3E83A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 8 kB, 17 pF input capacitance	-
MF3E93A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 16 kB, 17 pF input capacitance	-
MF3EH23A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 2 kB, 70 pF input capacitance	-
MF3EH43A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 4 kB, 70 pF input capacitance	-
MF3EH83A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 8 kB, 70 pF input capacitance	-
MF3EH93A1DUF/01EV	FFC	12 inch wafer (sawn; 75 µm thickness)[1][2]; 16 kB, 70 pF input capacitance	-
MF3E23A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 2 kB, 17 pF input capacitance	SOT500-4
MF3E43A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 4 kB, 17 pF input capacitance	SOT500-4

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07794-2ac4-4a3e-90e-6ec63cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 3. Ordering information for MIFARE DUOX for EV-Charging...*continued*

Type number	Package		
	Name	Description	Version
MF3E83A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 8 kB, 17 pF input capacitance	SOT500-4
MF3E93A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 16 kB, 17 pF input capacitance	SOT500-4
MF3EH23A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 2 kB, 70 pF input capacitance	SOT500-4
MF3EH43A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 4 kB, 70 pF input capacitance	SOT500-4
MF3EH83A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 8 kB, 70 pF input capacitance	SOT500-4
MF3EH93A0DA8/01EV	MOA8	plastic leadless module carrier package[3]; 16 kB, 70 pF input capacitance	SOT500-4

[1] Delivered on film frame carrier with electronic fail die marking according to SECSII format.
[2] See [36]
[3] See Figure 99

Provided under NDA only
COMPANY CONFIDENTIAL - Legal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

5 Block diagram

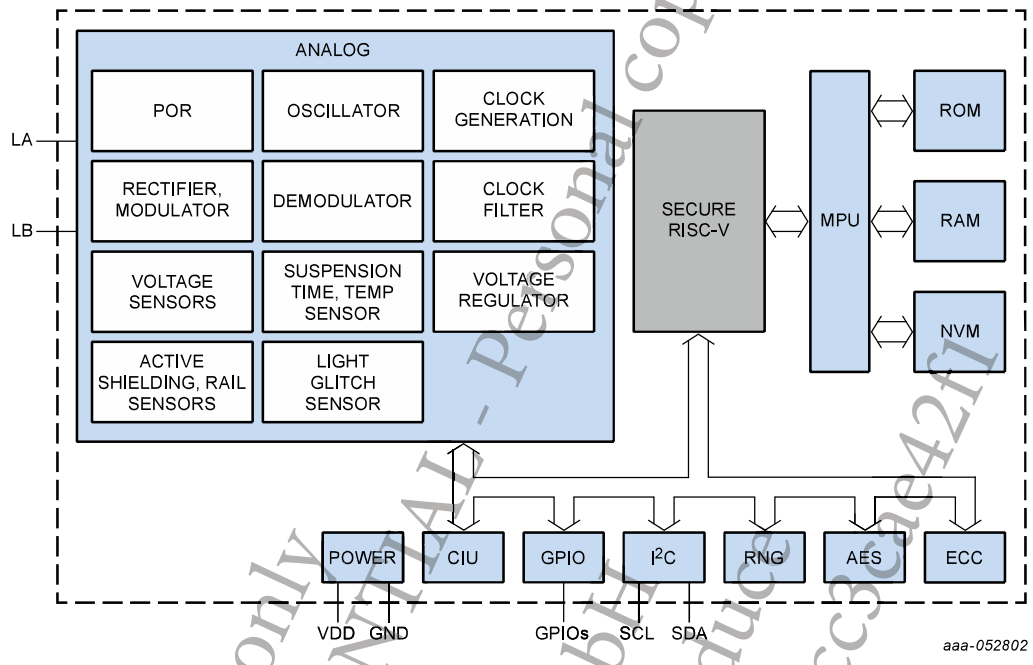


Figure 2. Block diagram

Provided under NDA only - Personal copy for:
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae421f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

6 Pinning information

The pinning information for the contactless smartcard module SOT500-4 (MOA8) is shown in Figure 3.

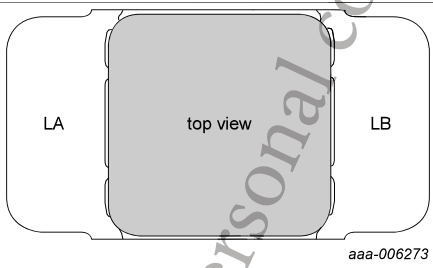


Figure 3. Contact assignments for SOT500-4 (MOA8 module)

The pin allocation is described in Table 4.

Table 4. Bonding pad assignment for SOT500-4 (MOA8 module)

Contactless Interface Module		MF3E(H)x3A0DA8
Antenna contacts	**Symbol**	**Description**
LA	LA	Antenna coil connection LA
LB	LB	Antenna coil connection LB

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc9cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7 Functional description

7.1 Introduction

MF3E(H)x3 is a contactless multiapplication smart card IC compliant with ISOIEC 14443A (part 1-4) and provides an off-the-shelf platform for smart card application providers.

MF3E(H)x3 builds on the feature set of MIFARE DESFire EV3. This document describes the full functionality, that is, including MIFARE DESFire functionality.

The memory organization of MF3E(H)x3 is flexible and can be dynamically structured to fit into any application requirements, like it is already known from the MIFARE DESFire EV3 product. Each application folder is a container of data files usable within a certain real-world application (for example, access management). There are five file types available for data storage and one file type for storing Transaction MAC as detailed in Section 7.12

Within the application folder, there is a set of keys, certificates, and configuration settings dedicated for the application. The application owner can freely organize the file structure and security settings within their application. An adjacent application will not have access to its files as long as they do not possess the correct security rights. The detailed description of application management is in Section 7.11.

MF3E(H)x3 also supports the ISO/IEC 7816-4 file structure and APDU as detailed in Section 7.17.

MF3E(H)x3 supports confidential and integrity-protected communication (see Section 7.5). The EV2 secure messaging (see Section 7.5.10), provides state-of-the-art crypto and security utilization.

Additionally, MF3E(H)x3 offers a transaction-oriented backup mechanism to prevent inconsistent updating of data storage across multiple files during a tearing situation. When transaction tearing occurs, either all data fields are updated or none is altered. The detailed description is in Section 7.14.

Besides the application file structure support, which is already well known from the MIFARE DESFire EV3 product, the product offers many new and enhanced features.

The following list gives an overview of the main new features, compared to MIFARE DESFire EV3, with pointers into the document:

- ECC-based Mutual and Reader-Unilateral Authentication, see Section 7.5.6.
- ECC-based Card-Unilateral Authentication, see Section 7.5.7. This is also used for Originality Checking purposes, see Section 7.21. Additionally generic ECDSA support is offered, see Section 7.15.
- The access right management has been extended for the ECC-based authentications. This is explained in Section 7.6.
- All certificate-related aspects for the ECC-based authentications are collected in Section 7.10.
- All asymmetric key management aspects for the ECC keys are collected in Section 7.9.
- Other existing symmetric features have been extended with an ECC-based equivalent:
  - Transaction MAC with the Transaction Signature, as described in Section 7.14.4.
  - Secure Dynamic Messaging with an ECC-based signature, as described in Section 7.5.11, especially Section 7.5.11.1.10 and Section 7.5.11.1.11.
- Symmetric crypto features have been extended with AES-256 support:
  - AES-based symmetric authentication, see Section 7.5.8.
  - AES-based symmetric secure messaging (EV2 Secure Messaging), see Section 7.5.10.
  - Secure Dynamic Messaging, see Section 7.5.11.
  - The symmetric key management, see Section 7.8, and especially Section 7.8.6.1 and Section 7.8.6.2.
  - Transaction MAC, see Section 7.14.3.
  - Proximity Check, see Section 7.18.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Gerhard Hoffmann
Do not copy, share or reproduce
Quantum Solutions GmbH
5fc07f9d-ca2c-405f-90fe-6ec0cae4ff1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

- I^2C support. The I^2C channel itself is defined in [3], but some implications on access right management can be found in Section 7.6.
- GPIO support, see Section 7.16.
- EV charging support, see Section 7.19. This is supporting the command set as defined by VDE-AR-E 2532-100.
- SetConfiguration has been extended for the new features. GetConfiguration is introduced to allow retrieval of the configurations.

The following features are not supported on MF3E(H)x3 compared to MIFARE DESFire EV3:

- D40 and EV1 authentication and secure messaging. This includes the ISO/IEC 7816-4 based authentication. From the latter, only ISOGetChallenge is supported as a generic random source. This means that for symmetric authentication only EV2 authentication and secure messaging is supported. The EV2 secure messaging is also used as the secure channel after an ECC-based mutual or reader-unilateral authentication.
- The AES-based Originality Check and the Originality Check based on a static ECDS signature have been replaced by an ECC-based challenge-response protocol.
- The privacy preserving Virtual Card Selection method and related AuthVCMandatory and AuthPCMandatory configurations.
- MIFARE Classic support. The MIFARE Classic mapping as supported by MIFARE DESFire EV3 is not supported.
- At delivery the KeyID.PICCMasterKey is configured as a KeyType.AES128 key instead of a 2TDEA key.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3ca4f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.2 Terminology

7.2.1 Normative statements and informative comment

The command API is described in tables and figures, see Section 8.

The tables are normative. For some commands, the protocol is described in formulas. If this is the case, this is also normative.

7.2.2 Notation in formulas and pseudo code

In the description of the protocol the following notation is used:

Table 5. Notation in formulas and pseudo code

Symbol	Example	Description
→	→	Message flowing from the left-hand entity to the right-hand entity.
←	←	Message flowing from the right-hand entity to the left-hand entity.
=	$A = B\|\|C$	Assignment of the value of $B\|\|C$ to A.
E	E(K,X)	Symmetric encryption of plaintext X using secret key K.
D	D(K,X)	Symmetric decryption of ciphertext X using secret key K.
MAC	MAC(K,M)	Message authentication code of message M using secret key K.
MACt	MACt(K,M)	Truncated message authentication code of message M using secret key K.
||	$x\|\|y$	x concatenated with y
. . .	$x_1\|\| \ldots \|\|x_n$	x_1 concatenated with all messages $x2$, etc. up until x_n.
[n..m]	X[15..8]	Bytes m to n of the preceding byte string, so in the example of X. Bytes within a byte string are numbered from least significant/rightmost to most significant/leftmost starting with index 0 for the least significant. There holds m ≤ n. When used for transmitted messages, the byte string is handed to the layer that implements the transmission. The order how bytes and their bits are transmitted is defined in the [5] and [6] standard and is out of scope for this specification. Example MAC(Km, Msg) [15 .. 8] Take the 8 most significant bytes of the 16-byte MAC value in the same order as they are in the MAC.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 5. Notation in formulas and pseudo code*...continued*

Symbol	Example	Description
[]	[x]	Within the command API definition and formulas, the square brackets can have another meaning. They indicate optional parameters.
[b n..m]	X[b127..64]	Bits m to n of the preceding byte string, so in the example of X. Bits within a byte string are numbered from the least significant bit in the least significant/ rightmost byte to the most significant bit in the most significant/leftmost byte. There holds m ≤n. Example MAC (Km, Msg) [b 127 .. 64] This is the same as MAC (Km, Msg) [15 .. 8] and takes the leftmost 64 bits from the MAC of 16 byte (= 128 bit)
a.. b		Range of values from a to b, both inclusive.
⌊..⌋	⌊*a*/16⌋	Floor function. The largest integer that is smaller than or equal to a/16.
⌈..⌉	⌈*a*/16⌉	Ceiling function. The smallest integer that is larger than or equal to a/16.
!=	if A <> B then	Not equal.
==	if A == B then	Equal.
Random()	A:=Random(16)	Draw a random number of 16 bytes long.
“”		Empty string.
Null	A:= Null	Nothing. The variable A will not hold a value after the assignment.
Len( )	Len(a)	Length of the string a in number of bytes.

Adjust(Str, Length)	Result:=Adjust(S, L)	Take the Length most significant bytes of Str; pad if necessary.
		If Len(S) > L then
		Result := S [Len(S)-1..Len(S)-L]
		else
		Result := Pad(S,L)
		endif

Pad(x, n)	Pad(PCDcap2, 6)	Pad the end of the byte string x with zero bytes (0x00) until the string is n bytes in length. If the byte string is already n bytes or more, the result is x.
AND	if a=1 AND b=2	Logical and condition.
OR	if a=1 OR b=2	Logical or condition.
$x \oplus y$		XOR of bit-vectors x and y.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 5. Notation in formulas and pseudo code*...continued*

Symbol	Example	Description
X <<< n	RndA <<< 1	left-rotation of the byte-vector X by n byte positions.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.3 ISO/IEC 14443 support

MF3E(H)x3 supports ISO/IEC 14443-3 and ISO/IEC 14443-4 of PICC Type A. Only the default parameter values will be specified here. The complete specification can be found in ISO/IEC 14443-3 [5] and ISO/IEC 14443-4 documents [6] .

Higher bit rates than 848 kbps and related S(PARAMETERS) are supported to enable PICC to PCD communication of up to 3.4 MBit/s.

7.3.1 Default parameter values

7.3.1.1 ATQA

ATQA as defined in ISO/IEC 14443-3. The default value is *0x0344* for the 7-byte UID. It is transmitted least significant byte (LSB) first as *0x44 0x03*.

ATQA reflects by the default the UID size which is by default set to the 7-byte double size UID. However MF3E(H)x3 supports Random IDs of single size (4 bytes). This feature can be activated with SetConfiguration as specified in Section 7.7.3.1. If the Random ID feature is enabled, then the ATQA is changed to *0x0304*. It is transmitted least significant byte (LSB) first as *0x04 0x03* as specified in ISO/IEC14443-3. When enalbing Random ID or NUID with SetConfiguration, the size indication in the ATQA will be adapted automatically.

In case of 10-byte UID, the default value is 0x384, transmitted 0x84 0x03.

The ATQA values can be modified using the command SetConfiguration as specified in Section 7.7.3.1.

7.3.1.2 SAK

SAK as defined in ISO/IEC 14443-3. The default value SAK1 is *0x24* (indicating UID not complete). The value SAK2 is *0x20* (indicating UID complete). The SAK values can be modified using the command SetConfiguration as specified in Section 7.7.3.1.

7.3.1.3 ATS

ATS as defined in ISO/IEC 14443-4. The default value is as follows:

- *TL = 0x06*
- *T0 = 0x78*
- *TA(1) = 0x77*
- *TB(1) = 0x81*
- *TC(1) = 0x02*
- *T1 = 0x80*

T0 = 0x78 sets by default the FSC to 256 bytes. This is the maximum supported size. Note that MF3E(H)x3 supports also smaller frame sizes, e.g. 64 byte being backwards compatible with MIFARE DESFire. This can be set using the command SetConfiguration as specified in Section 7.7.3.1. If the frame size is changed, then the T0 value has to be adapted according to ISO/IEC 14443-4 (for example, *T0 = 0x78* for 256 bytes).

As encoded by TA(1) = 0x77, MF3E(H)x3 supports next to the standard 106 kbit/s bit rate also higher bit rates: 242 kbit/s, 424 kbit/s, and 848 kbit/s. The PICC is able to handle different bit rates in different direction.

Note: FWI value (in *TB(1)*) smaller than 7 is not supported by this platform and shall never be used.

7.3.1.4 UID

The UID default value is set at the production phase and follows the ISO/IEC 14443-3 specification.

Provided under NDA only. COMPANY CONFIDENTIAL - Personal copy for: Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc07792-522c-405c-90fe-6ecc3cae42f1

7.4 I²C support

In specific product types / product configurations of MF3E(H)x3, an optional I²C interface is available and can be used.

Please refer to Section 2.2, for the availablility of the I²C interface in different package types.

If the I²C interface is available in the product configuration, following points apply:

MF3E(H)x3 supports I²C target communication with 7-bit target address according to [31].

The following bus speeds are supported, though potentially limited by pull-up resistance and load capacitance depending on the HW configuration.

- 100 KHz (Standard-mode)
- 400 KHz (Fast-mode)
- 1 MHz (Fast-mode Plus)

Clock stretching is not supported.

At the data link layer, the T=1' protocol as specified in [20] is supported. Only the default parameter values will be specified here.

MF3E(H)x3 supports arbitration between the ISO/IEC 144443 and I²C interface, by serving the first interface where activity is detected. Further communication interfaces transfer is supported as specified in Section 7.16.1.

7.4.1 Default I²C parameter values

7.4.1.1 Target address

The default target address shall be 0x20. The target address can be changed through SetConfiguration Option 0x10, see Section 7.7.3.1.

7.4.1.2 Communication interface parameters

The communication interface parameters (CIP) as defined by [20] are specified in Table 6.

Table 6. I²C communication interface parameters

Name	Length	Description	Value
PVER	1	Protocol Version: [20] defines version '01' of the protocol.	0x01
Length of IIN	1	Length of Issuer Identification Number	0x04
IIN	3-4	Issuer Identification Number (according to [9], BCD encoded)	0x63070093
PLD	1	Physical Layer ID: '01' for SPI / '02' for I²C	0x02
Length of PLP	1	Length of Physical Layer Parameters	0x08
Configuration	1	Characteristics supported by SE: b1 = 0: Clock stretching not supported b1 = 1: Clock stretching supported Other bits: RFU	0x00
PWT	1	Power Wake-up Time (ms)	0x02
MCF	2	Maximal Clock Frequency at which the SE may operate (in kHz)	0x03E8 (1MHz)
PST	1	Power Saving Time-outs (in ms)	0xFF

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 6. I^2C communication interface parameters*...continued*

Name	Length	Description	Value
MPOT	1	Minimum Polling Time (conditional to Polling Mode support) (in ms)	0x01
RWGT	2	R/W Guard Time (in us)	0x0064
Length of DLLP	1	Length of Data Link Layer Parameters	0x04
BWT	2	Block Waiting Time (in ms)	0x03E8 (ca. 1 sec)
IFSC	2	Maximum Information Field Size of the SE (in bytes) (that is, initial value)	0x00FE
Length of HB	1	Length of Historical Bytes (Max. 32 bytes)	0x00
Historical Bytes	Var	Empty	-

7.4.2 I2C Application Remarks

7.4.2.1 Power Management

MF3E(H)x3 contains an adaptive power management system reducing or stopping internal clocks in case of internal voltage drops occurring in weak field conditions when powered via the NFC interface or during switching of the VCC.

In case the internal clock is stopped MF3E(H)x3 might not be able to serve the I2C bus while the internal clock is stopped. In this case the host will read 'FF' while the internal clock is stopped.

This cases will be detected with high probability by the CRC check.

The recommended error-recovery on failed CRC checks is as following:

1. Read IFSC number of bytes to clear before continuing.

2. Send R-Block CRC Fail to Card

In case the length information bytes are read as FF the host protocol stack shall abort reading after the maximum frame size (254+6 bytes) supported by MF3E(H)x3, e.g. by checking if the response is longer than IFS + protocol overhead.

7.4.2.2 Write after Write behavior

For Write after Write with two correct transmit messages the device response is discarded when the new message is received. Instead of the expected read message an error message A5-82-00-00-89-E0 (Other Error).

7.4.2.3 Waiting Time Extension behavior

The default block waiting time is defined as 300ms and shall not be changed. Changed block waiting times may lead to missing Waiting Time Extension (WTX) interrupts in case NFC and I2C communication.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.5 Authentication and Secure Messaging

MF3E(H)x3 supports two classes of commands: a native command set and a command set according to ISO/IEC 7816-4 [11], which includes both wrapping of the native commands and a subset of the standard ISO/IEC 7816-4 commands. The native command set and the related secure messaging are defined in this and the following chapters, while the ISO/IEC 7816-4 command set is defined in Section 7.17.

7.5.1 Native Command Format

One can distinguish several parts in the native command and response APDU format supported by MF3E(H)x3.

A command as sent by the PCD consists of the concatenation of:

- the command code (Cmd)
- zero, one or more header fields (CmdHeader)
- zero, one or more data fields (CmdData)

The response as sent by the PICC consists of the concatenation of:

- the return code (RC)
- zero, one or more data fields (RespData)

This is illustrated in Figure 4. Note that the colors used in this figure will be used in the table definitions of the command set to distinguish the different parts. These parts will be used to define generically the secure messaging. Optional fields are shown with dashed lines and square brackets ([...]) around the byte count.

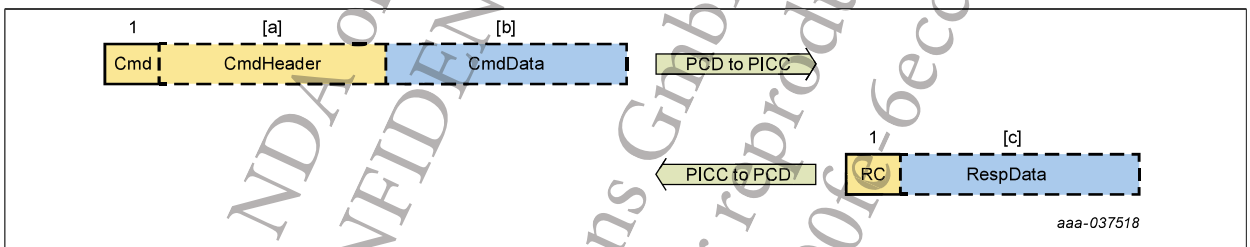


Figure 4. Native command format

Next to this native command format, MF3E(H)x3 supports the APDU message structure according to ISO/IEC 7816-4 [11] for:

- wrapping of the native command format into a proprietary ISO/IEC 7816-4 APDU
- a subset of the standard 7816-4 commands

A detailed description of the supported ISO/IEC 7816-4 wrapping and standard commands is specified in Section 7.17.

7.5.1.1 Byte Order

On the native command interface, plain command parameters consisting of multiple bytes are represented least significant byte (LSB) first, similar as for ISO/IEC 14443 parameters [5] during the activation.

For cryptographical parameters and keys (including the random numbers exchanged during authentication, the TI and the computed MACs), this does not hold. For these, the representation on the interface maps one-to-one to the most significant byte (MSB) first notation used in this specification, see also Section 7.2.

Note that within this document, the '0x' prefix indicates hexadecimal integer notation, that is, not reflecting the byte order representation on the command interface at all.

Provided under NDA only. COMPANY CONFIDENTIAL - Personal copy for: Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-405c-90fc-6ecc3cae42f1

7.5.2 Command Chaining and Multiple Pass Commands

MF3E(H)x3 supports two kinds of chaining mechanisms to overcome the FSD/FSC limitations [6]:

- The standard ISO/IEC 14443-4 command chaining [6], as further defined in Section 7.5.2.1.
- A native command chaining mechanism, as further defined in Section 7.5.2.2.

7.5.2.1 ISO/IEC 14443-4 Chaining

ISO/IEC 14443-4 chaining, as defined in [6], is supported in the following cases:

- a PCD can always split a native command in several frames, as long as the length of the reassembled INF field remains within the supported buffer size of 60 bytes.
- additionally, the PICC supports ISO/IEC 14443-4 chaining to allow for larger command or response frames than 256 bytes for variants of the following commands:
  - Native commands ReadData with *Cmd* 0xAD, WriteData with *Cmd* 0x8D, ReadRecords with *Cmd* 0xAB, WriteRecord with *Cmd* 0x8B and UpdateRecord with *Cmd* 0xBA.
  - Native commands wrapped into ISO/IEC 7816-4 APDU: ReadData with *Cmd* 0xAD, WriteData with *Cmd* 0x8D, ReadRecords with *Cmd* 0xAB, WriteRecord with *Cmd* 0x8B and UpdateRecord with *Cmd* 0xBA, see Section 7.17.2.
  - Standard ISO/IEC 7816-4 commands: ISOReadBinary, ISOUpdateBinary, ISOReadRecord, and ISOAppendRecord, that is, every command where larger frame size can occur.
- the PICC will automatically split a response in several frames to fit with the FSD frame size supported by the PCD.

Note that the native commands listed above use a different command code if using ISO/IEC 14443-4 chaining, than when using native chaining, see Section 7.13.

When a PCD applies ISO/IEC 14443-4 chaining, it needs to assure the reassembled INF field (see [6]) containing the command header (that is, ISO/IEC 7816-4 header bytes and/or (Cmd||CmdHeader)) fits within the PICC's buffer (FSC). If not, the PICC may respond with LENGTH_ERROR.

The ISO/IEC 14443-4 chaining does not influence the secure messaging. This means the secure messaging mechanisms are applied as if the command or response would have been sent in a single large frame. With regard to command execution, commands are handled as if they were received in one large frame, except for write commands where the total frame size can be larger than the supported FSC (WriteData, WriteRecord, UpdateRecord, ISOUpdateBinary, and ISOAppendRecord). In this case, an implementation is already allowed to start command execution before the complete command is received.

Note that this is important for FileType.StandardData files, where the file may end up in an undefined state if the command is not completed successfully.

7.5.2.2 Native Chaining

For some commands, MF3E(H)x3 offers a native chaining mechanism to support command execution where the complete command or response does not fit into a single native frame, or to support security protocol execution, that is, if the complete command exists of multiple passes between PCD and PICC (for example, authentication commands).

This section does not provide a complete overview but defines the general principles. See individual command definition for which command supports the native chaining.

The following fundamental states are introduced:

- CmdState.NoCmdInProgress: this is the default state where no command is in progress.
- CmdState.CmdInProgress: this is the state where a command is in progress: native chaining is applied.

Provided under NDA only
CONFIDENTIAL
Seagate Data Solutions GmbH
Quantum IT Solutions GmbH
Do not copy, share, reproduce
5fc077bd-02c-4033-9fb6-ec3cae211

The PICC will return ADDITIONAL_FRAME if native chaining is needed. Depending on the command, this means the PICC expects additional command data or has additional response data to be sent. The PICC is in CmdState.CmdInProgress and the current command (CurrCmd) is remembered.

After having responded with ADDITIONAL_FRAME, the PICC expects the next command frame to start with command code 0xAF to continue the ongoing command. If another command code is received this will be rejected with COMMAND_ABORTED. After this, the chaining is aborted and the PICC accepts new commands.

As soon as an RC different from ADDITIONAL_FRAME is sent, the PICC is in CmdState.NoCmdInProgress and any command can be accepted (depending on the access rights, authentication state, . . . see later).

In CmdState.NoCmdInProgress, that is, if native chaining is not active, the chaining command code 0xAF will not be accepted, and responded with ILLEGAL_COMMAND_CODE.

If secure messaging is active, the secure messaging is applied as if the command or response would have been sent in a single large frame. The 0xAF command and response code are ignored for these calculations. If a MAC is appended to the command or response, this may result in an additional frame if necessary.

With regards to command execution, commands are handled as if they were received in one large frame, except for write commands where the total frame size can be larger than the supported buffer size (WriteData, WriteRecord, and UpdateRecord). In this case, an implementation is already allowed to start command execution before the complete command is received.

Note that this is important for FileType.StandardData files, where the file may end up in an undefined state if the command is not completed successfully.

7.5.3 Error Handling

In case of unsuccessful command execution, MF3E(H)x3 will send a return code different from OPERATION_OK or ADDITIONAL_FRAME. Next to this, the PICC will also execute the following abort actions:

- Abort command: the currently ongoing command (see Section 7.5.2) is aborted and a new command is expected. The PICC moves into CmdState.NoCmdInProgress.
- Abort transaction: the currently ongoing transaction is aborted. This is similar as the execution of AbortTransaction, see Section 7.14.2.2.
- Abort secure messaging: the currently active secure messaging is aborted and the authentication state is lost, see also Section 7.5.5. The PICC moves into VCState.DFNotAuthenticated.

The complete list of error code can be found in Section 8.1.

The following generic error cases can occur. The reason for this error is identical and therefore not further detailed as is done for other errors in the command tables in Section 8.

- COMMAND_ABORTED: An ongoing chained command or multiple-pass command is aborted, see Section 7.5.2.2.
- ILLEGAL_COMMAND_CODE: The received command code (Cmd) is not supported.
- LENGTH_ERROR: The command size is not allowed according to the definition in Section 8 or due to the supported buffer size.
- PARAMETER_ERROR: Parameter value is not allowed according to the definition in Section 8. This can happen for RFU settings or parameters with restricted input range.

The secure messaging can also be aborted, losing the authentication state, by:

- SelectApplication or ISOSelectFile: selecting an application, even if the same application is reselected.
- DeleteApplication: deleting the current application.
- ChangeKey or ChangeKeyEV2: if changing the key which was used for the currently active authentication.
- RollKeySet: changing the AKS.
- ISOGeneralAuthenticate, AuthenticateEV2First or AuthenticateEV2NonFirst: starting a new authentication.

Provided under NDA only
COMPANY CONFIDENTIAL - personal copy for:
Sergi Santa
Quantum IT solutions GmbH
Do not copy, share or reproduce
5fc07794-22c-445c-90fe-6eccc3cae2f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

- PreparePC, ProximityCheck, VerifyPC: failure during proximity check.
- ISO/IEC 14443-4 DESELECT or power off

7.5.4 Communication Modes

MF3E(H)x3 supports three communication modes as defined in Table 7. As shown in the table, the different communication modes are represented by two bits. This representation is used at several places in the document.

Table 7. Supported communication modes

Communication mode	Bit Representation	Explanation
CommMode.Plain	X0b	No protection: message is transmitted in clear
CommMode.MAC	01b	MAC protection for integrity and authenticity
CommMode.Full	11b	Full protection for integrity, authenticity and confidentiality

The communication mode defines the level of security for the communication between PCD and PICC.

At application and PICC level, the communication mode is defined by the command itself, as specified in Table 8. The specified communication mode is applied if there is an active authentication regardless of whether this authentication is required by the command or not.

At file level, the communication mode is defined by the file. The specified communication mode is applied if there is an active authentication. Note however that, under an active authentication, if the only valid access condition for a certain access right is free access (0xE), see also Section 7.12.2, CommMode.Plain is to be applied. Note that this exception only applies to data management commands, see Section 7.13 and ChangeFileSettings, see Section 7.12.2.3. For other commands, where the communication mode is not configurable, the default is applied always when authenticated.

The commands for authentication and changing keys have their own secure messaging rules, as indicated by N/A (not applicable) in the table. As SelectApplication resets any active authentication, no communication mode is applicable for this command as well.

If there is no active authentication, the command and response are sent in plain (or the command is rejected in the case an authentication is required).

Table 8. Overview of expected command communication mode for Secure Messaging

Command	Communication mode
AbortTransaction	CommMode.MAC
AuthenticateEV2First	N/A (command specific)
AuthenticateEV2Non First	N/A (command specific)
ChangeFileSettings	CommMode.Full
ChangeKey	N/A (command specific)
ChangeKeyEV2	N/A (command specific)
ChangeKeySettings	CommMode.Full
ClearRecordFile	CommMode.MAC
CommitReaderID	CommMode.MAC
CommitTransaction	CommMode.MAC
CreateApplication	CommMode.MAC

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 8. Overview of expected command communication mode for Secure Messaging...*continued*

Command	Communication mode
CreateBackupDataFile	CommMode.MAC
CreateCyclicRecordFile	CommMode.MAC
CreateDelegated Application	CommMode.Full
CreateLinearRecord File	CommMode.MAC
CreateStdDataFile	CommMode.MAC
CreateTransaction MACFile	CommMode.Full
CreateValueFile	CommMode.MAC
Credit	CommMode of targeted file.
CryptoRequest	CommMode of CryptoRequest as defined by SetConfiguration 0x15.
Debit	CommMode of targeted file.
DeleteApplication	CommMode.MAC
DeleteFile	CommMode.MAC
ExportKey	CommMode of targeted file.
FinalizeKeySet	CommMode.MAC
Format	CommMode.MAC
FreeMem	CommMode.MAC
GetApplicationIDs	CommMode.MAC
GetCardUID	CommMode.Full
GetConfiguration	CommMode.Full
GetDFNames	CommMode.MAC
GetDelegatedInfo	CommMode.MAC
GetFileCounters	CommMode.Full
GetFileIDs	CommMode.MAC
GetFileSettings	CommMode.MAC
GetISOFileIDs	CommMode.MAC
GetKeySettings	CommMode.MAC
GetKeyVersion	CommMode.MAC
GetValue	CommMode of targeted file.
GetVersion	CommMode.MAC
InitializeKeySet	CommMode.MAC
LimitedCredit	CommMode of targeted file.
ManageCARootKey	CommMode of targeted key, or if targeting not yet existing key CommMode.Full.
ManageGPIO	CommMode of ManageGPIO as defined by SetConfiguration 0x11.
ManageKeyPair	CommMode of targeted key, or if targeting not yet existing key CommMode.Full.

Provided under NDA with COMPANY CONFIDENTIAL - Personal copy for: Sergi Santa Quantum IT Solutions GmbH Do not copy, share or reproduce 5fc079d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 8. Overview of expected command communication mode for Secure Messaging...*continued*

Command	Communication mode
ReadData	CommMode of targeted file.
ReadGPIO	CommMode of ReadGPIO as defined by SetConfiguration 0x11.
ReadRecords	CommMode of targeted file.
RollKeySet	CommMode.MAC
SelectApplication	N/A
SetConfiguration	CommMode.Full
UpdateRecord	CommMode of targeted file.
WriteData	CommMode of targeted file.
WriteRecord	CommMode of targeted file.

Note that the communication mode is closely, but not 1-to-1 related with whether a command requires an authentication. For example, whether GetKeySettings requires an authentication depends on a PICC or application level setting. This is defined in the detailed command descriptions. A similar behavior is true for file access, where the requirement for an authentication will depend on the file access right conditions, see also Section 7.12.2.

7.5.5 State Diagram

The fundamental states as listed below are introduced in Figure 5. AuthKey is a variable maintained to remember the key currently authenticated with. Depending on this key, access rights to subsequent commands may be granted or not.

- VCState.NotAuthenticated: this is the default state where there is no active authentication. The AuthKey is invalidated in this state, and also no CertAccessRights are granted. This state is reached after ISO/IEC 14443-4 activation, both from ISO/IEC 14443-3 HALT and IDLE state.
- VCState.PartiallyAuthenticated: in this state an authentication is ongoing. The PICC is expecting the second part. This means any previous active authentication has already been lost.
- VCState.AuthenticatedAES: there is an active authentication reached by successfully executing the symmetric authentication protocol initiated with AuthenticateEV2First or AuthenticateEV2NonFirst. EV2 Secure Messaging, as defined in Section 7.5.10, is active. The targeted key of the last authentication is remembered as AuthKey. Depending on this key, access rights to subsequent commands may be granted or not.
- VCState.AuthenticatedECC: there is an active authentication reached by successfully executing the asymmetric authentication protocol initiated with ISOGeneralAuthenticate and ISOGeneralAuthenticatePart2, or ISOSelectFile (if initiating integrated asymmetric authentication) and ISOGeneralAuthenticateFinal. Also here, symmetric AES-based EV2 Secure Messaging, as defined in Section 7.5.10, is active. Access rights in this state are defined as CertAccessRights, and depend on the targeted KeyID.CARootKey and/or reader certificates presented during the authentication, see Section 7.6.3 and Section 7.6.4.

The transitions to and from those states are related to the secure messaging specification. Access control for other commands is based on the key used for reaching the authenticated state.

Provided under NDA only
COMPANY CONFIDENTIAL
Sergii Sagatov, Quantum Solutions GmbH
Do not copy, share or reproduce
Personal copy for:
5fc0779d-22c-4096-90fe-6ecc3ca8a2f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

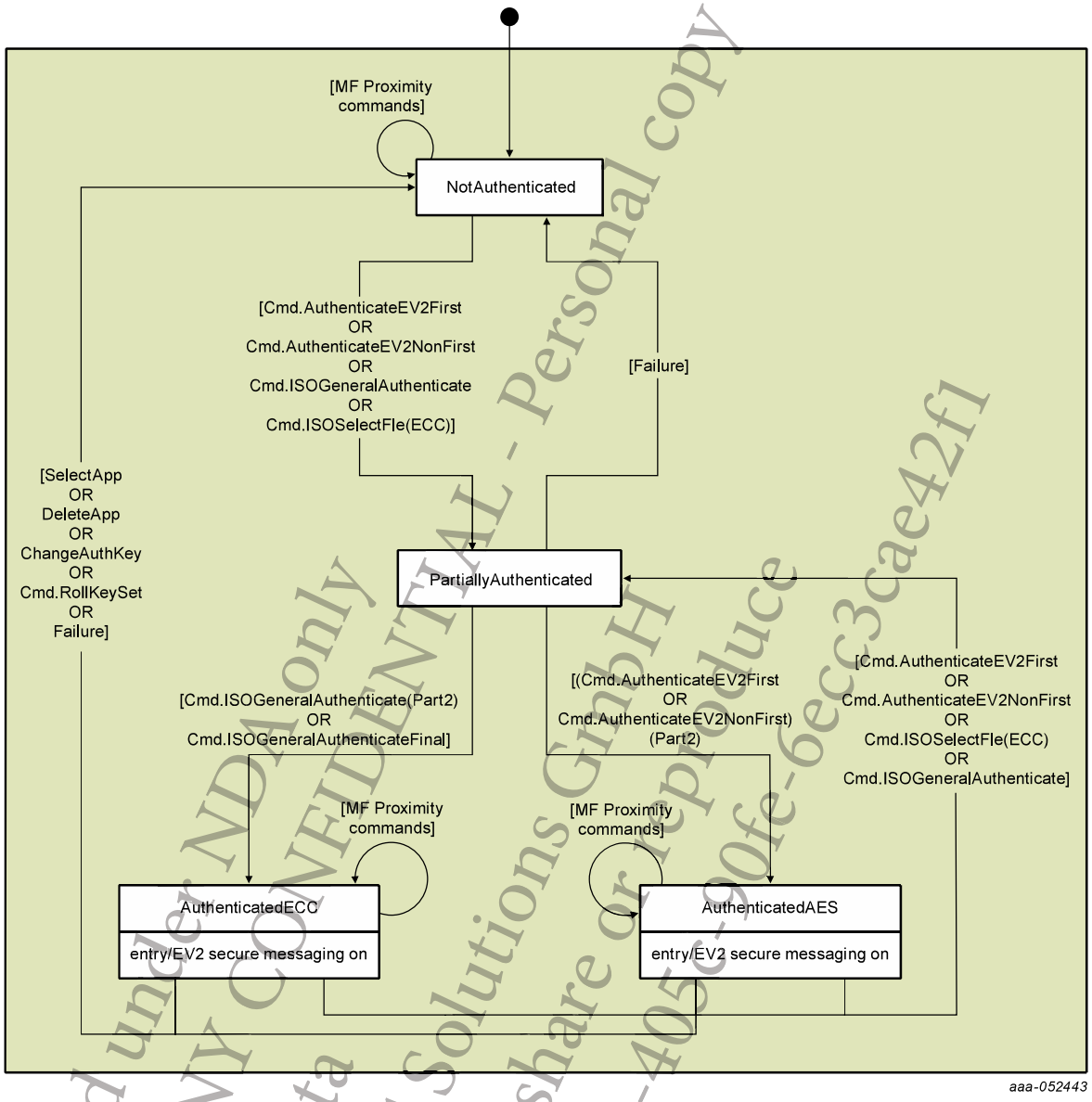


Figure 5. Authentication State Diagram

Here are the notes for Figure 5:

- *Failure* indicates any error as defined in Section 7.5.3: the PICC switches to the VCState.NotAuthenticated. Note that in these cases the response is already sent with CommMode.Plain (as always in VCState.NotAuthenticated).
- The authentication process consists of two parts. If only the first part is received, then any command different from the expected second part will result in a failure. ISO/IEC 14443-4 Deselect is allowed at any time.
- In both, VCState.AuthenticatedAES and VCState.AuthenticatedECC the same AES-based secure messaging applies. Note however that in VCState.AuthenticatedAES either AES-128 or AES-256 is applied, depending on the key type of the key targeted during the preceding authentication. In VCState. AuthenticatedECC, always AES-128 is applied, which is from an equivalent security strength as the 256-bit ECC applied during the authentication.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

- In VCState.NotAuthenticated, KeyID.VCProximityKey is used for proximity checking (that is, for VerifyPC) in this case. In VCState.AuthenticatedAES and VCState.AuthenticatedECC, KeyID.SesAuthMACKey is used for VerifyPC. A Proximity command received in the wrong sequence generates a *Failure* event.
- *SelectApp* indicates application selection via SelectApplication or ISOSelectFile.
- *ChangeAuthKey* indicates ChangeKey or ChangeKeyEV2 targeting the currently authenticated key. Note that this only applies when changing symmetric keys during a symmetric authentication.
- *DeleteApp* indicates DeleteApplication targeting the currently selected application. In these cases the authentication is also lost.

7.5.6 ECC-based Mutual and Reader-Unilateral Authentication

MF3E(H)x3 supports an ECC-based mutual authentication protocol as described in this section. The protocol can be executed via two command sequences:

- ISOGeneralAuthenticate and ISOGeneralAuthenticatePart2
- ISOSelectFile and ISOGeneralAuthenticateFinal: the authentication is initiated by and integrated with the application selection with ISOSelectFile. This is only possible if the targeted application is configured for this, see Section 7.5.6.6.

On top, also a reduced version of the protocol is supported where only the reader (or PCD) will authenticate itself: ECC-based Reader-Unilateral Authentication. This allows for delaying the card authentication till the end of the transaction, for example, via an ECC Transaction Signature (see Section 7.14.4). This has the advantage of saving some ECC operations, but should only be applied after carefully assessing the system security, as the read and write operations are not authenticated by the card at the moment of their execution, that is, a reader has no guarantee it is communicating with a valid card at that point in time.

7.5.6.1 Data structures and notations

7.5.6.1.1 ECC Key Pair

MF3E(H)x3 supports ECC key pairs over the following curves:

- Curve P-256, as defined by NIST in FIPS 186-4 [4].
- Curve brainpoolP256r1, as defined in RFC 5639 [21]

The following notations will be used:

- (*Priv.ID, Pub.ID*): a static key pair, consisting of a private and a public key with *ID* being the identity of the key owner. Typically, this will be a long-term key pair from which the private key *Priv.ID* is only known to the owner. The public key *Pub.ID* is exchanged during the protocol to the other party, ensuring its authenticity via means of a certificate.
- (*E.Priv.ID, E.Pub.ID*): an ephemeral key pair, consisting of a private and a public key with ID being the identity of the key owner. This key pair is generated at the start of the protocol and the public key E.Pub.ID is exchanged to the other party.
- (*Priv.CA.ID, Pub.CA.ID*): a static key pair used by a Certificate Authority (CA) to sign certificates. Here *ID* is the identity of the party engaged in the protocol whose certificate or certificate chain is issued by this CA. Note that potentially, the CA and its key pair can be the same for both parties. The *Pub.CA.ID* must be provisioned to the other party before the execution of the protocol to allow this party to validate the certificate (chain) received during the protocol.
- (*Pub.ID.x, Pub.ID.y*): individual coordinates of the public key being a point on the elliptic curve. Unless otherwise stated, on the API public keys will be in uncompressed point representation, i.e. *0x04*||*Pub.ID.x*||*Pub.ID.y* .

In the following, *A* will be used for the reader/PCD identity and *B* for the card/PICC identity.

Protected under NDA CONFIDENTIAL Personal copy for: Quantum Secure Solutions GmbH, Do not copy, share or reproduce 5fc07b5-22c-405c-90fe-6ecc2c4ef1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.5.6.1.2 Certificate

During the protocol, the authenticity of the public key to be used to authenticate the other party, will be guaranteed via a certificate or certificate chain.

For the card side, MF3E(H)x3 is in general agnostic of the certificate format used, see Section 7.10.1. The trust-provisioned certificates that are stored on the card during production are detailed in Section 7.10.1.4.

For the reader side, MF3E(H)x3 supports certificate chains of up to two certificates. In case of a single certificate, the authenticity of the public key *Pub.A* is guaranteed by the certificate's signature. The signature is generated with (*Priv.CA.A* and can be verified with *Pub.CA.A*). In case of chaining the parent certificate is signed by the CA with *Priv.CA.A*, while the leaf certificate holding *Pub.A* is signed with the key pair of the parent certificate.

The reader certificate format is further detailed in Section 7.10.2.

The following notations will be used:

- *Cert.A*: the certificate (in case of single certificate) or certificate chain, that is used to authenticate the *Pub.A* used by the party referred to by *A*.
- *Cert.A.parent*: the parent certificate.
- *Cert.A.leaf*: the leaf certificate.

For a reader certificate chain, the chain is the concatenation of the parent and leaf certificate, that is, *Cert.A* = *Cert.A.parent*||*Cert.A.leaf*.

Note that *Cert.A* may also be omitted.

7.5.6.2 Cryptographic primitives

7.5.6.2.1 Elliptic Curve Diffie-Hellman

MF3E(H)x3 supports key agreement with ephemeral keys implementing the Elliptic Curve Cryptography Cofactor Diffie-Hellman primitive as defined in NIST SP800-56A [26].

The following notation will be used:

Z = ECDH(E.Priv.A, E.Pub.B)

In the above example, *A* combines his private key *E.Priv.A* with the public key *E.Pub.B* received from *B* to compute the shared secret *Z*. The size of *Z* equals the curve size, that is, 32 bytes for the curves supported by MF3E(H)x3. The domain parameters are implicitly known from the targeted static keys.

7.5.6.2.2 Elliptic Curve Digital Signature Generation and Verification

The supported ECDSA algorithm is defined in Section 7.5.7.2.1 .

7.5.6.2.3 MAC Calculation

MAC calculation, as required for the session key generation of the ECC authentication, is done according to the CMAC standard described in NIST Special Publication 800-38B [3] using AES-128 as the underlying block cipher according to [1]. Padding and IVs are applied according to the standard [3]. No truncation is to be applied.

The notation *MAC(key, message)* is used to denote the MAC calculation.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07790-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.5.6.2.4 Encryption

Encryption and decryption, as applied during the ECC authentication, are calculated using the underlying block cipher according to the CBC mode of NIST SP800-38A [2].

Padding is applied according to Padding Method 2 of ISO/IEC 9797-1 [12], that is, by adding always 0x80 followed, if required, by zero bytes until a string with a length of a multiple of 16 byte is obtained. Note that if the plain data is a multiple of 16 bytes already, an additional padding block is added.

There is one exception: in case of unilateral authentication, Msg.B.enc is constructed with random padding. For the encryption during the ECC authentication, the IV will be 128 bits of 0.

The notation *E(key, message)* is used to denote the encryption and *D(key, message)* for decryption.

7.5.6.3 Authentication overview

The ECC-based authentication supported by MF3E(H)x3 is based on the well-known Station-to-Station protocol for authenticated key agreement, as originally introduced in [34].

An overview of this asymmetric mutual authentication between PICC and PCD is given in Table 9. The notations and cryptographic primitives have been defined in the previous sections, see Section 7.5.6.1 and Section 7.5.6.2. The session key generation *(KDF(. . .))* is specified in Section 7.5.6.4.

Note that the actual mapping of message components to commands may slightly vary depending on the actual variant, as further specified in Section 7.5.6.6 and Section 7.5.6.5.

The protocol provides authenticated key agreement with perfect forward secrecy. Compared to the original protocol from [34], this variant adds mutual entity authentication via certificate-based signatures. Identity misbinding attacks as for example, presented in [22], where one party incorrectly believes he has established a key with another party, are prevented by including the certificates in the encrypted data, protected by the shared secret, relying on the non-malleability property of the CBC-encryption. Manipulating the encrypted data will change the plaintext related to the manipulated ciphertext block in uncontrollable way, and therefore will be detected via the signature and certificate validation. Encrypting the certificates also provides additional privacy properties, as the identity of both parties are protected against passive eavesdropping. On top, the identity of the PICC is even protected against active adversaries, as the PICC will only disclose its identity if the PCD successfully authenticated.

Table 9. ECC-based mutual authentication

PCD	PICC
Knows: *Priv.A, Cert.A, Pub.CA.B* The PCD generates an ephemeral key pair: *(E.Priv.A; E.Pub.A)*	**Knows**: *Priv.B, Cert.B, Pub.CA.A*
Opts.A || *E.Pub.A* →	
	The PICC generates an ephemeral key pair: *(E.Priv.B, E.Pub.B)* The PICC computes the shared secret: *ShS = ECDH(E.Priv.B, E.Pub.A)* The PICC derives the session keys: $K_{SesAuthENC}$, $K_{SesAuthMAC}$ = *KDF(ShS)*
E.Pub.B ←	

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-5ecc3cae42f1

Table 9. ECC-based mutual authentication *...continued*

PCD	PICC
The PCD computes the shared secret: *ShS = ECDH(E.Priv.A, E.Pub.B)* The PCD derives the session keys: $K_{SesAuthENC}$, $K_{SesAuthMAC}$ = *KDF(ShS)* The PCD computes its signature: *Msg.A.ext = 0xE0E0 || OptsA || E.Pub.A || E.Pub.B* *Sig.A =* $ECDSA_{Sign}$*(Priv.A, Msg.A.ext)* The PCD encrypts its message: *Msg.A.pl = 0xE0E0 || Cert.A || Sig.A* *Msg.A.enc =* $E(K_{SesAuthENC}$*,Msg.A.pl)*	
Msg.A.enc →	
	The PICC decrypts the message: *0xE0E0 || Cert.A || Sig.A =* $D(K_{SesAuthENC}$*, Msg.A.enc)* The PICC validates the certificate (chain) with *Pub.CA.A*: this includes one or two $ECDSA_{Verify}$. The PICC validates the signature: *Msg.A.ext = 0xE0E0 || OptsA || E.Pub.A || E.Pub.B* $ECDSA_{Verify}$*(Pub.A, Msg.A.ext)* The PICC computes its signature: *Msg.B.ext = 0xE1E1 || OptsB || E.Pub.B || E.Pub.A* *Sig.B =* $ECDSA_{Sign}$*(Priv.B, Msg.B.ext)* The PICC encrypts its message: *Msg.B.pl = 0xE1E1 || OptsB || Cert.B || Sig.B* *Msg.B.enc =* $E(K_{SesAuthENC}$*, Msg.B.pl)* In case of successful execution, the PICC enters VCState.AuthenticatedECC, applying EV2 secure messaging for subsequent commands and granting the access rights encoded in the received certificate.
Msg.B.enc ←	
The PCD decrypts the message: *0xE1E1 || Opts.B || Cert.B || Sig.B =* $D(K_{SesAuthENC}$*, Msg.B.enc)* The PCD validates the certificate (chain) with *Pub.CA.B*: this includes one or more $ECDSA_{Verify}$. The PCD validates the signature: *Msg.B.ext = 0xE1E1 || OptsB || E.Pub.B || E.Pub.A* $ECDSA_{Verify}$*(Pub.B, Msg.B.ext)* In case of successful execution, the PCD enters the authenticated state, applying EV2 secure messaging for subsequent commands.	

The above table spells out the mutual authentication. For the reader-unilateral authentication variant, the differences are limited:

- The PICC does not have a static key pair, that is, no *Priv.B*.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Seigii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

- The PICC does not present a certificate, that is, no *Cert.B*.
- The PICC does not compute a signature, that is, no *Sig.B*.
- The PICC computes *Msg.B.enc* as follows:
  - *Msg.B.pl = 0xE1E1 || OptsB*
  - *Msg.B.enc = E($K_{SesAuthENC}$, Msg.B.pl)*

Both the mutual and unilateral authentication variants are mapped to the same command APDUs. MF3E(H)x3 supports two variants. For the detailed command description, see Section 7.5.6.5 and Section 7.5.6.6.

7.5.6.4 Session Key Generation

During the ECC authentication, both the PICC and the PCD generate two AES-128 session keys: KeyID.SesAuthMACKey and KeyID.SesAuthENCKey. The KeyID.SesAuthENCKey is already used during the authentication itself. Both keys are further on used for the secure messaging after a successful authentication, as defined in Section 7.5.10.

The session key generation for the ECC authentication is according to the Two-Step Key Derivation of NIST SP800-56C [27].

The first step ensures proper randomness extraction, by applying AES-128 CMAC, see Section 7.5.6.2.3, to calculate a key-derivation key as follows:

KDK = MAC(E.Pub.A.x[7..0] || E.Pub.B.x[7..0], ShS)

As key for the MAC calculation a salt consisting of the concatenation of the 8 least significant bytes of the x-coordinates of the exchanged ephemeral public keys is used for additional randomization. The message input consists of the shared secret ShS which was the result of the ephemeral key agreement.

The second step uses this key-derivation key KDK for key expansion. This key expansion is done according to NIST SP 800-108 [23] in counter mode.

The pseudo random function *PRF(key, message)* is again AES-128 CMAC, see Section 7.5.6.2.3.

The input data is constructed using the following fields as defined by [23]. Note that NIST SP 800-108 allows defining a different order than proposed by the standard as long as it is unambiguously defined.

- a 2-byte label, distinguishing the purpose of the key: 0x4BB4 for MACing and 0xB44B for encryption
- a 2-byte counter, fixed to 0x0001 as only 128-bit keys are generated.
- a 2-byte length, fixed to 0x0080 as only 128-bit keys are generated.
- a 10-byte context, set to all 0x00

Therefore, the 16-byte session keys are constructed as follows:

$K_{SesAuthENC}$ *= MAC(KDK, 0xB4||0x4B||0x00||0x01||0x00||0x80||0x00||0x00||0x00||0x00||0x00||0x00||0x00||0x00||0x00||0x00)*

$K_{SesAuthMAC}$ *= MAC(KDK, 0x4B||0xB4||0x00||0x01||0x00||0x80||0x00||0x00||0x00||0x00||0x00||0x00||0x00||0x00||0x00||0x00)*

Note thus that for VCState.AuthenticatedECC, always AES-128 session keys are generated as the security level of AES-128 is equivalent to the 256-bit ECC applied during the authentication.

7.5.6.5 ISOGeneralAuthenticate and ISOGeneralAuthenticatePart2

This section defines the ECC-based mutual or reader-unilateral authentication which is initiated with ISOGeneralAuthenticate.

The authentication consists of two parts ISOGeneralAuthenticate and ISOGeneralAuthenticatePart2. Detailed command definition can be found in Section 8.2.1 and Figure 25.

Provided under NDA only. COMPANY CONFIDENTIAL - Personal copy for: Sergi Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5f0779d-c22c-405c-90fe-6ecc3cae42f1

The protocol cannot be interrupted by other commands. On any command different from ISOGeneralAuthenticatePart2 received after the successful execution of the first part, the PICC aborts the ongoing authentication.

This is also the case if the certificate file is not related with a valid private key, or the targeted public and private key belong to different elliptic curves. The CA root key (Pub.CA.A) will authenticate the PCD by validating the certificate (chain) received in the second step, while the targeted certificate (Cert.B) and related PICC private key (Priv:B) will authenticate the PICC.

All parameters in the command and response datafield are BER-TLV data objects (DOs) encoded according to ISO/IEC 7816-4 [11] with DER length encoding. Authentication DOs are collected under the 0x7C tag according to ISO/IEC 7816-4, Table 100. Other parameters use a context specific tag according to ISO/IEC 8825-1 [14]. All DOs must be sent in the order specified in the command tables.

Upon reception of ISOGeneralAuthenticate, the PICC will check the keys and certificates addressed in P2.

The parameter OptsA is a TLV-structure with Tag 0x80 consisting of the following fields:

- OptsA.Value[0]: AuthMethod
  - Bit 7-6
    - '10': mutual authentication
    - '01': unilateral authentication
  - Bit 5
    - '0': Cert.A present
    - '1': Cert.A omitted
  - Bit 4-0: RFU
- OptsA.Value[1]: ProtocolVersion = 0x00
- OptsA.Value[2]: CertFileNo (only present if mutual authentication)
- OptsA.Value[3]: PrivKeyNo (only present if mutual authentication)

As this OptsA parameter may be used for potential future extensions (see Section 7.20), the PICC accepts TLVstructures with a bigger length. Additional bytes in the value field will be ignored by the PICC.

ProtocolVersion shall be set to 0x00. Bigger versions are reserved for future use, but will be accepted by MF3E(H)x3, returning a specific success return code 0x9Fxx with xx the maximal supported version, that is, 0x9F00 for MF3E(H)x3. The protocol will be further executed, allowing the reader to seamless fall back to that version.

If the targeted public key (*CARootKeyNo*), or if applicable, private key (*PrivKeyNo*) or certificate file (*CertFileNo*) do not exist, an error is returned. This is also the case if the targeted public and private key belong to different elliptic curves. If the targeted KeyID.ECCPrivateKey is not configured for ECC-based Mutual Authentication, or has an enabled KeyUsageCtrLimit that was already reached, see Section 7.9.1.2, the command is also rejected.

The KeyID.CARootKey (*Pub.CA.A*, addressed by *CARootKeyNo* via P2 Bit 2-0) will authenticate the PCD by validating the certificate (chain), or in case certificates are omitted, directly the signature, as received in the second step. If applicable, that is, if the targeted KeyID.CARootKey is associated with a active CRLFile, revocation checks will be done during the authentication, see Section 7.10.2.3.

In case multiple application selection, the KeyID.CARootKey addressed by *CARootKeyNo* is called the primary KeyID.CARootKey and is located in the primary application. In that case, also a second CA root key is addressed (*secCARootKeyNo* via P2 Bit 6-4). This key is located in the secondary application. The key value and associated issuer name (if any) of that KeyID.CARootKey must be the same as for the primary addressed KeyID.CARootKey. This means also either both must have an issuer name associated, or none. On top, if one of them is associated with a CRLFile for certification revocation purposes, the other must also be associated with the same CRLFile.

Provided under NDA only
CONFIDENTIAL - Permanent copy for:
Scott Infotech GmbH
Quantum IT Solutions
Do not copy, share or reproduce
5fc07f1-2-2c-4b5c-90e-6ecc3cae42f1

As a result of the certificate validation, the PICC retrieves the access rights CertAccessRights which will be granted during this authentication session, see Section 7.6.4. These access rights are limited by the access rights associated by the targeted key, see Section 7.6.4. In case of multiple application selection, the access rights of the primary key are used for CertAccessRights of the primary application, while the access rights of secondary KeyID.CARootKey are used for the CertAccessRights of the secondary application.

If the access rights, granted by the targeted and certificate chain, are not sufficient to obtain FileAR.Read or FileAR.ReadWrite for the targeted *CertFileNo*, the command will as well be rejected. The targeted certificate (*Cert.B*, addressed by *CertFileNo*) and related PICC private key (*Priv.B*, addressed by *PrivKeyNo*), in case of mutual authentication, will allow to authenticate the PICC.

Second or shared application addressing via SAI, as introduced in Section 7.11.3.1, is supported for *CertFileNo*. This means for a targeted KeyID.ECCPrivateKey which must reside in the primary application, this can be complemented with an application-specific certificate that is located in the secondary application.

The received ephemeral key *E.Pub.A* is validated by the PICC, executing a partial public key validation according to NIST SP800-56A [26] on the same curve as the keys targeted by P2. As all supported curves have co-factor=1, no full public key validation is required.

Once the command is successfully parsed and validated, the PICC will generate its own ephemeral key pair (*E.Priv.B, E.Pub.B*), also on the same curve. The public key will be returned as *E.Pub.B*. Both parties can now compute the shared secret. This is done by the PICC by calculating *ShS = ECDH(E.Priv.B, E.Pub.A)*. Also the session keys KeyID.SesAuthMACKey and KeyID.SesAuthENCKey are generated according to Section 7.5.8.4.

Upon reception of ISOGeneralAuthenticatePart2, the PICC will decrypt Msg.A.enc using KeyID.SesAuthENCKey, retrieving 0x3030[|| *Cert.A*] || *Sig.A*. *Cert.A*, if present, consists of either a single certificate, or a certificate chain of up to two certificates, as defined in Section 7.10.2.1. The PICC will validate the certificate chain according to the procedure defined in Section 7.10.2.2 using the targeted *CARootKey*. Individual certificates of up to 440 byte are supported. This means that in case of a certificate chain, Cert.A can be up to 880 byte. Therefore, ISOGeneralAuthenticatePart2 shall support up to 960 byte for the length L2 and 964 byte for the length L1 mentioned in Table 6, that is, including the signature, a certificate chain and encryption padding.

The PICC uses the public key *Pub.A* retrieved from *Cert.A* (i.e. *Cert.A.leaf* in case of certificate chain) to verify the received signature *Sig.A*. If *Cert.A* is omitted the targeted *CARootKey* will be used to directly verify *Sig.A*. In both cases, this involves an ECDSA signature verification over the message: 0xE0E0 || *OptsA* || *E.Pub.A* || *E.Pub.B*. *OptsA* is the full received TLV-structure, while the ephemeral public keys are in uncompressed point representation, that is, all exactly as exchanged over the command interface.

In case of successful certificate validation and signature verifications, the PICC will respond with ISO9000, returning the following encrypted data field:

- if mutual authentication: $E(K_{SesAuthENC}, 0xE1E1 \| OptsB \| Sig.B \| Cert.B)$
- if reader-unilateral authentication: $E(K_{SesAuthENC}, 0xE1E1 \| OptsB)$

The parameter OptsB is reserved for future extensions, (see Section 7.20). It is an LV-structure, holding the PDCap2: *OptsB* = 0x06 || *PDCap2*.

The PICC signature is computed as follows: *Sig.B = ECDSASign(Priv.B, 0xE1E1* || *OptsB* || *E.Pub.B* || *E.Pub.A)*. *Cert.B* is the *CertificateData* of *CertificateLength* as encoded in the file targeted by *CertFileNo*, see Section 7.10.1.

On successful execution, the PICC enters VCState.AuthenticatedECC and the EV2 Secure Messaging is activated, see Section 7.5.10. The already computed session keys are applied: KeyID.SesAuthMACKey and KeyID.SesAuthENCKey. The CmdCtr is reset to zero, and also the TI is set to all zero bytes for EV2 Secure Messaging in VCState.AuthenticatedECC. On any failure during the protocol the PICC ends up in VCState.NotAuthenticated.

Provided under NDA by NXP - Personal copy for: COMPANY CONFIDENTIAL - Registrierte Quantum IT Solutions GmbH Do not copy, share or reproduce 5fc07f0d-22c4-405c-afe6-6e3ca42f

7.5.6.6 ISOGeneralAuthenticateFinal initiated by ISOSelectFile

To allow further transaction optimization, the ECC-based authentication can also be initiated during application selection with ISOSelectFile. To support this the targeted application must be configured to support this with SetConfiguration Option 0x04, see Section 7.7.3.1, while the ECC-based Mutual and Reader-Unilateral Authentication must be enabled for the targeted interface, as configured with SetConfiguration Option 0x0F for NFC and SetConfiguration Option 0x10 for I^2C. This can also be supported at PICC-level. If supported, the ephemeral public key of the PICC (*E.Pub.B*) is already returned via the FCI in the ISOSelectFile response.

The authentication consists of two parts ISOSelectFile and ISOGeneralAuthenticateFinal. Detailed command definition can be found in in Section 8.12.1 and Figure 85, and in Section 8.2.2 and Figure 26.

The processing is very similar as for the authentication initiated with ISOGeneralAuthenticate, and therefore not fully described in detail again. Only the differences are highlighted:

- The public key of the freshly generated ephemeral key pair of the PICC *E.Pub.B* is encoded in the FCI response from ISOSelectFile. The whole FCI consists of: 0x6F || 0x44 || 0x85 || 0x41 || E.Pub.B with E.Pub.B in uncompressed point representation.
- The targeted KeyID.CARootKey (via *CARootKeyNo*), and KeyID.ECCPrivateKey (via *PrivKeyNo*) with its certificate file (via *CertFileNo*) are only provided on the second command exchange, that is, via P2. The targeted public key and private key must belong to the same curve. This curve has been pre-configured with SetConfiguration Option 0x04, see Section 7.7.3.1, when enabling ECC-based authentication via ISOSelectFile for the targeted application. In case of multiple application selection, also a secondary KeyID.CARootKey is addressed (via *secCARootKeyNo*).
- The parameter OptsA is only provided on the second command exchange, that is, ISOGeneralAuthenticateFinal.
- The public key of the ephemeral key pair of the PCD E.Pub.A is only provided on the second command exchange, that is, ISOGeneralAuthenticateFinal. It is embedded in the Authentication Data Objects TLVstructure together with Msg.A.enc.

For the rest, the processing is exactly the same as what is described in Section 7.5.6.5.

7.5.7 ECC-based Card-Unilateral Authentication

MF3E(H)x3 supports an ECC-based card-unilateral authentication protocol as described in this section. This allows for authenticating the card without requiring an authentication from the reader side. This protocol can be applied for Originality Check purposes, that is, to ensure genuineness of MF3E(H)x3 ICs, as described in Section 7.21.

The protocol can be executed with ISOInternalAuthenticate.

As the protocol creates a trace that cannot be repudiated the privacy implications of enabling the feature should be evaluated.

7.5.7.1 Data structures and notations

7.5.7.1.1 ECC Key Pair

The card-unilateral authentication will apply a static key pair (*Priv.B,Pub.B*) from which the private key *Priv.B* is stored on the card and used by the card during the protocol.

7.5.7.1.2 Certificate

For the protocol, the public key *Pub.B* to be used by the reader for validating the authenticity of the card, should be authenticated through a certificate or certificate chain. This certificate (chain) can be stored on

Provided under NDA only
COMPANY CONFIDENTIAL – Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

the card in a FileType.StandardData file and retrieved via the related commands before executing the ISOInternalAuthenticate.

Note that for Originality Check purposes, the certificate is trust-provisioned during manufacturing, as described in Section 7.22.1.

During the further description of the protocol the certificate validation is kept out of scope.

7.5.7.2 Cryptographic primitives

7.5.7.2.1 Elliptic Curve Digital Signature Generation and Verification

The card-unilateral authentication is based on the ECDSA Digital Signature Generation and Verification as defined in [28]. The hash function to be applied is SHA-256, as specified in NIST FIPS 180-4 [25].

The following notations will be used:

- $Sig.B = ECDSA_{Sign}(Priv.B, M)$
- $[true, false] = ECDSA_{Verify}(Pub.B, M, Sig.B)$

In the above example, *B* signs the message *M* with his private key *Priv.B*, resulting in the signature *Sig.B*.*Sig.B* consists of two integers (*Sig.B.r*,*Sig.B.s*) of a size equalling the curve size, i.e.both 32 bytes for an ECC-256 curve, resulting in a 64-byte signature. With $ECDSA_{Verify}$, the *Sig.B* is verified to be correct for the message *M* with the public key *Pub.B*, resulting in *true* or *false*.

7.5.7.3 Authentication overview

The ECC-based card-unilateral authentication supported by MF3E(H)x3 is based on the two-pass unilateral authentication as standardized in ISO/IEC 9798-3 [29].

An overview of this asymmetric card-unilateral authentication is given in Table 10.

Note that the inclusion of a random number generated by the card prevents the reader from having full control on the data that gets “signed” by the card. This is different from a generic ECDSA signature generation as supported with CryptoRequest_ECCSign.

Table 10. ECC-basedcard-unilateral authentication

PCD	PICC
Knows: *Pub.B* The PCD generates a random challenge *RndA*	**Knows:***Priv.B*
OptsA || *RndA* →	
	The PICC generates a random *RndB*: The PICC computes its signature: $Sig.B = ECDSA_{Sign}(Priv.B, 0xF0F0[\ \|\|\ OptsA]\ \|\|\ RndB\ \|\|\ RndA)$
RndB || *Sig.B* ←	
ThePCD validates the signature: $ECDSA_{Verify}(Pub.B, 0xF0F0[\ \|\|\ OptsA]\ \|\|\ RndB\ \|\|\ RndA, Sig.B)$	

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.5.7.4 ISOInternalAuthenticate

The authentication is initiated by ISOInternalAuthenticate. A detailed command definition can be found in in Section 8.2.3 and Figure 27.

The protocol can only be executed in VCState.NotAuthenticated and does not change the authentication state. All parameters in the command and response data field are BER-TLV data objects (DOs) encoded according to ISO/IEC7816-4 [11] with DER length encoding. Authentication DOs are collected under the 0x7C tag according to ISO/IEC 7816-4, Table 100. Other parameters use a context-specific tag according to ISO/IEC 8825-1 [30]. All DOs must be sent in the order specified in the command tables.

Upon reception of ISOInternalAuthenticate, the PICC checks the KeyID.ECCPrivateKey addressed by P2 if the key does not exist or is not enabled for ECC-based unilateral authentication, the command is rejected. If the targeted KeyID.ECCPrivateKey has an enabled KeyUsageCtrLimit that was already reached, see Section 7.9.1.2, the command is also rejected.

MF3E(H)x3 supports ISOInternalAuthenticate at the PICC level by default for originality checking with the Priv.Orig at KeyNo 0x01. If preferred, for example, for privacy reasons, this can be disabled by updating the key. For the Priv.PICC at KeyNo 0x00, the protocol is not supported by default, but could be enabled through the key management.

At the application level, it depends on the key policy configured during key creation or update, whether the protocol is supported for a specific key.

The parameter OptsA is optional. As it may be used for potential future extensions (see Section 7.20), the current implementation will accept and ignore it, including TLV-structures with a bigger length. If present, *OptsA* is included in the signature calculation to allow protection against future protocol downgrade attacks. Future implementations may then also return an *OptsB* with Tag 0x80.

Upon reception of the command, the PICC generates an own 16-byte random number *RndB* and create the signature as follows: *Sig.B = ECDSASign*(*Priv.B,*0x*F*0*F*0[|| *OptsA*] || *RndB* || *RndA*). Note that for *OptsA* the full TLV-structure is included, while for RndA and RndB only the 16-byte random values are included.

7.5.8 AES-based Symmetric Authentication

The AES-based symmetric authentication is managed by AuthenticateEV2First and AuthenticateEV2NonFirst. This authentication was introduced in MIFARE DESFire EV2, explaining the command naming.

7.5.8.1 Authentication overview

Prior to data transmission applying secure messaging, a symmetric mutual three pass authentication can be done between the PICC and PCD.

An overview of this symmetric mutual three pass authentication between PICC and PCD is given in Table 11.

Table 11. Three pass authentication

PCD	PICC
The reader device is always the entity which starts an authentication procedure. This is done by sending an authentication command. As parameter the key number is passed to the PICC in order to select a certain key stored in its non-volatile memory. Depending on the application selection (and possibly the presence of the SAI, see Section 7.11.3.1, the subsequent authentication procedure is done for the targeted application (or for the PICC level).)	
Authenticate(KeyNo)	

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ec23cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 11. Three pass authentication...*continued*

PCD	PICC
	The PICC generates a random challenge *RndB*. This random number is enciphered with the selected key K_x.
$E(K_x$, RndB)	
The PCD deciphers the received message and retrieves *RndB*. (The used key for the deciphering obviously has to be the same as for the previous enciphering by the PICC.) The *RndB* is rotated by eight bits (first byte is moved to the end of *RndB*), yielding *RndB*'. The PCD itself generates a random challenge *RndA*. This random number is enciphered with the selected key K_x. The PCD enciphers this *RndA* concatenated with the rotated *RndB*' using the selected K_x.	
$E(K_x$, RndA || RndB')	
	The PICC deciphers the received message and retrieves *RndA* and *RndB*'. The PICC now verifies the sent *RndB* by comparing it against the received *RndB*' after rotating the original *RndB* left by eight bits. A successful verification proves to the PICC that the PICC and the PCD possess the same secret key. If the verification fails, the PICC stops the authentication procedure and returns an error message. The *RndA* is rotated by eight bits (first byte is moved to the end of *RndA*), yielding *RndA*' which is enciphered again by the PICC. In case of successful execution, the PICC enters the related authentication state and generates the session key(s).
$E(K_x$, RndA')	
The PCD deciphers the received message and retrieves *RndA*'. The PCD now verifies the sent *RndA* by comparing it against the received *RndA*' after rotating the original *RndA* left by eight bits. A successful verification proves to the PCD that the PICC and the PCD posses the same secret key. If the verification fails, the PCD exits the authentication procedure and may halt the PICC.	
In case of successful execution, the PCD enters the related authentication state and generates the session key(s).	In case of successful execution, the PICC enters the related authentication state and generates the session key(s).

Note that depending on the actual authentication commands, some of the message contains additional parameters not shown in the table above, as they are related to the actual secure messaging mode.

In the remainder, there is made mention of First Authentication and Non-First Authentication. A First Authentication is done with AuthenticateEV2First. The Non-First Authentication is done with AuthenticateEV2NonFirst and can only be applied after a First Authentication, that is, in state VCState.AuthenticatedAES. Correct application of AuthenticateEV2First and AuthenticateEV2NonFirst allows cryptographically binding all messages within a transaction by using a transaction identifier, see Section 7.5.10.1, and a command counter, see Section 7.5.10.2, even if multiple authentications are required.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Qualitätssicherung IT Solutions GmbH
Document copy, state of 2025-06-26
5fc07791-22c-405c-901c-cce42f1

The following Table 12 specifies when to authenticate using AuthenticateEV2First and when to use AuthenticateEV2NonFirst.

Table 12. When to use which authentication command

Purpose	AuthenticateEV2First	AuthenticateEV2NonFirst
First authentication (that is, when not in VCState.AuthenticatedAES)	Allowed	Not Allowed
Subsequent authentication (that is, when in VCState.AuthenticatedAES)	Allowed, recommended not to use	Allowed, recommended to use

Note that it is possible to use AuthenticateEV2First in state VCState.AuthenticatedAES. This can be used if the PCD does not care about interleaving attacks but rather prefers a simpler implementation. Note that the messages of the ongoing transaction are then not bound cryptographically anymore. Using AuthenticateEV2First followed by AuthenticateEV2NonFirst is recommended therefore. In this way an attacker cannot let a PICC work with two PCDs at the same time and in that way compromise the security.

7.5.8.2 AuthenticateEV2First Command

This section defines the EV2 Authentication, which is mandatory to be used first in a transaction when using EV2 Secure Messaging initiated by symmetric mutual authentication. In this procedure both, the PICC as well as the PCD show in an encrypted way that they possess the same secret, that is, the same key. This authentication is supported with KeyType.AES128 or KeyType.AES256 keys.

The authentication consists of two parts: AuthenticateEV2FirstPart2. Detailed command definition can be found in Section 8.2.4 and Figure 28.

The protocol cannot be interrupted by other commands. On any command different from AuthenticateEV2FirstPart2 received after the successful execution of the first part, the PICC aborts the ongoing authentication.

During this authentication phase, the PICC must accept messages from the PCD that are longer than the lengths derived from this specification as long as *LenCap* is correct. This requirement is to support the upgradeability features (see Section 7.20) on future product versions. The current content of *PCDcap2* shall not be interpreted by the PICC. The PCD will reject answers from the PICC when they do not have the proper length.

Upon reception of AuthenticateEV2First, the PICC will validate the targeted key. If the key does not exist, AuthenticateEV2First is rejected. Within an application, this depends on the number of symmetric application keys. At PICC level (AID = 0x000000), the available keys are defined in Table 32. The command is rejected if the targeted key is not of KeyType.AES128 or KeyType.AES256. At PICC level, the command is rejected if the key cannot be used for authentication or has not been enabled yet, see Section 7.8.4.1. If the SAI is set and no second application is selected, the command is rejected.

The PICC will generate a random 16-byte challenge *RndB* and send this encrypted to the PCD, according to Section 7.5.10.4. Additionally, the PICC will reset CmdCtr to zero and generate a random TI.

Upon reception of the AuthenticateEV2First response from the PICC, the PCD also generates a random 16-byte challenge *RndA*. The PCD encrypts, on his turn, the concatenation of *RndA* with *RndB'*, which is the received challenge after decryption and rotating left by one byte. Within AuthenticateEV2FirstPart2, this is sent to the PICC.

Upon reception of AuthenticateEV2FirstPart2, the PICC decrypts the second message and validates the received *RndB'*. If not as expected, the command is rejected. Else it will generate *RndA'* by rotating left the received *RndA*. This is returned together with the generated TI. Also, the PICC will send 12 bytes of capabilities to the PCD: 6 bytes of PCD capabilities *PCDcap2* that were received on the command (sent back for verification) and 6 bytes of PICC capabilities *PDcap2*.

Provided under NDA only - Personal copy for: CONFIDENTIAL - Rohde & Schwarz GmbH Copyright Satisfiers, Do not copy, share or reproduce 5f6007f79d-522c-4fcc-90fe-6ecc3ce42f

The use of those capabilities, and the negotiation process is described in Section 7.20. Note that part of PDCap will be configurable with SetConfiguration. PDCap2.2 reflects the configured TransactionTimer. *PDCap2* is used to refer both to the value sent from the PCD to the PICC and to the value used in the encrypted response message from the PICC to the PCD where in this case the PCDcap2 is the adjusted version of the originally sent *PCDcap2*: that is, truncated or padded with zero bytes to a length of 6 bytes if needed.

On successful execution of the authentication protocol, the session keys KeyID.SesAuthMACKey and KeyID.SesAuthENCKey generated according to Section 7.5.8.4. The PICC is in VCState.AuthenticatedAES and the EV2 Secure Messaging is activated. On any failure during the protocol the PICC ends up in VCState.NotAuthenticated.

If there is a mismatch between the capabilities expected by the PCD and the capabilities presented by the PICC to the PCD (both the *PDcap2* and the echoed/adjusted *PCDcap2*), it is the responsibility of the PCD to take the proper actions based on the application the PCD is running. This decision is outside the scope of this specification.

7.5.8.3 AuthenticateEV2NonFirst Command

This section defines the EV2 Non-First Authentication, which is recommended to be used if EV2 Secure Messaging is already active in VCState.AuthenticatedAES. In this procedure both, the PICC as well as the PCD show in an encrypted way that they possess the same secret, that is, the same key. This authentication is supported with KeyType.AES128 or KeyType.AES256 keys.

The authentication consists of two parts: AuthenticateEV2NonFirst and AuthenticateEV2NonFirstPart2. Detailed command definition can be found in Section 8.2.5 and Figure 29.

This command is rejected if there is no active EV2 authentication, except for the following differences:

- No *PCDcap2* and *PDcap2* are exchanged and validated.
- TI is not reset and not exchanged.
- CmdCtr is not reset.

After successful authentication, the PICC remains in VCState.AuthenticatedAES. On any failure during the protocol, the PICC ends up in VCState.NotAuthenticated.

7.5.8.4 Session Key Generation

At the end of a valid authentication with AuthenticateEV2FirstPart2 or AuthenticateEV2NonFirst, both the PICC and the PCD generate two session keys for secure messaging, as shown in Figure 6:

- KeyID.SesAuthMACKey or $K_{SesAuthMAC}$ for MACing of messages
- KeyID.SesAuthENCKey or $K_{SesAuthENC}$ for encryption and decryption of messages

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

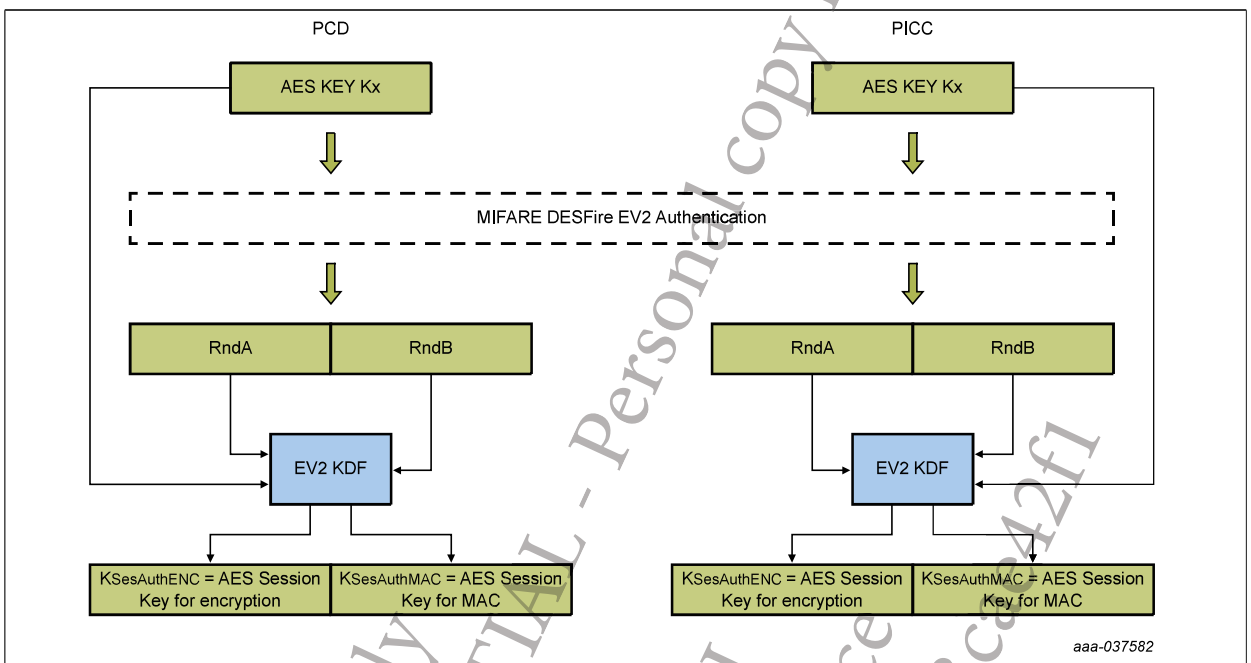


Figure 6. Session key generation for EV2 Secure Messaging

The session key generation is according to NIST SP 800-108 [23] in counter mode.

The pseudorandom function *PRF(key,message)* applied during the key generation is the CMAC algorithm described in NIST Special Publication 800-38B [3]. The key derivation key is the key K_x that was applied during authentication. If the authentication targets a KeyType.AES128 key, the generated session keys are also of KeyType.AES128. If a KeyType.AES256 authentication key is targeted, the session keys are also KeyType.AES256.

The input data is constructed using the following fields as defined by [23]. Note that NIST SP 800-108 allows defining a different order than proposed by the standard as long as it is unambiguously defined.

- a 2-byte label, distinguishing the purpose of the key: 0x5AA5 for MACing and 0xA55A for encryption
- a 2-byte counter
  - KeyType.AES128: fixed to 0x0001
  - KeyType.AES256: counting from 0x0001 to 0x0002
- a 2-byte length
  - KeyType.AES128: fixed to 0x0080
  - KeyType.AES256: fixed to 0x0100
- a 26-byte context, constructed using the two random numbers exchanged, *RndA* and *RndB*

7.5.8.4.1 KeyType.AES128

Firstly, the 32-byte input session vectors SV_x are derived as follows:

SV1 = 0xA5 || 0x5A || 0x00 || 0x01 || 0x00 || 0x80 || RndA[15..14] || (RndA[13..8] ⊕ RndB[15..10]) || RndB[9..0] || RndA[7..0]

SV2 = 0x5A || 0xA5 || 0x00 || 0x01 || 0x00 || 0x80 || RndA[15..14] || (RndA[13..8] ⊕ RndB[15..10]) || RndB[9..0] || RndA[7..0]

with ⊕ being the XOR-operator.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Then, the 16-byte session keys are constructed as follows:

$K_{SesAuthENC} = PRF(K_x, SV1)$
$K_{SesAuthMAC} = PRF(K_x, SV2)$

7.5.8.4.2 KeyType.AES256

Firstly, the 32-byte input session vectors SV_x are derived as follows:

$SV1a = 0xA5 \| 0x5A \| 0x00 \| 0x01 \| 0x01 \| 0x00 \| RndA[15..14] \| (RndA[13..8] \oplus RndB[15..10]) \| RndB[9..0] \| RndA[7..0]$

$SV1b = 0xA5 \| 0x5A \| 0x00 \| 0x02 \| 0x01 \| 0x00 \| RndA[15..14] \| (RndA[13..8] \oplus RndB[15..10]) \| RndB[9..0] \| RndA[7..0]$

$SV2a = 0x5A \| 0xA5 \| 0x00 \| 0x01 \| 0x01 \| 0x00 \| RndA[15..14] \| (RndA[13..8] \oplus RndB[15..10]) \| RndB[9..0] \| RndA[7..0]$

$SV2b = 0x5A \| 0xA5 \| 0x00 \| 0x02 \| 0x01 \| 0x00 \| RndA[15..14] \| (RndA[13..8] \oplus RndB[15..10]) \| RndB[9..0] \| RndA[7..0]$

with $\oplus$ being the XOR-operator.

Then, the 32-byte session keys are constructed as follows:

$K_{SesAuthENC} = PRF(K_x, SV1a) \| PRF(K_x, SV1b)$
$K_{SesAuthMAC} = PRF(K_x, SV2a) \| PRF(K_x, SV2b)$

7.5.9 Authentication Counter and Limit

To allow mitigating potential future attack scenarios, symmetric mutual authentications can be configured with a usage limitation. This allows limiting the amount of key computations, and therefore related trace collection for side-channel attacks. Next to attack mitigation, this feature can also be used to limit the usage of a card/device. Potentially, the limit can be increased in the field, for example, if the end-user pays for additional service.

The authentication usage limitation is configured through SetConfiguration Option 0x16.

Once enabled, MF3E(H)x3 shall maintain a AuthCtr for counting the authentications, and if configured also an AuthCtrLimit.

This means that the AuthCtr shall be incremented by the following operations:

- AuthenticateEV2First for AES-based authentication, before the response of the Part 1.

If the configured AuthCtrLimit has been reached, the related authentication will be disabled. This means that the relevant keys cannot be used anymore, though the key entry can still be updated (and potentially reenabled) if the required authentication to do so can still be gained, that is, through an asymmetric authentication if symmetric authentication is disabled.

If the AuthCtrLimit is disabled, authentications may still be counted.

When further updating SetConfiguration Option 0x16, it is possible to disable or change the AuthCtrLimit without affecting the current AuthCtr value. Note that putting the limit to a value equal or lower than the current value, will immediately disable the authentication.

As any other SetConfiguration option, the current authentication counter configuration and AuthCtrLimit can be retrieved with GetConfiguration. Note that for this Option 0x16, also the current AuthCtr value will be returned by GetConfiguration.

Enabling the feature may create a denial-of-service risk. It must be assessed from system-level perspective if this can be accepted.

Note that enabling the feature within an application consumes 32 byte of user memory.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sebastian Schmitz
Quantum Technologies GmbH
Do not copy, share or reproduce
55c077d-22c055-906-9ecc3cae4251

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.5.10 EV2 Secure Messaging

The EV2 Secure Messaging is an AES-based secure messaging which was introduced in MIFARE DESFire EV2, explaining the naming.

The EV2 secure messaging can both be initiated by a ECC-based authentication as defined in Section 7.5.6, as well as by AES-based authentication as defined in Section 7.5.8.

7.5.10.1 Transaction Identifier

MF3E(H)x3's transaction management is defined in Section 7.14. The EV2 Secure Messaging defines the concept of a Transaction Identifier to support cryptographically binding together all messages in all authentication sessions (if initiated by AuthenticateEV2First and followed by subsequent AuthenticateEV2NonFirst) that are used within the transaction.

In order to avoid interleaving of transactions from multiple PCDs toward one PICC, the Transaction Identifier TI must be included in each MAC that is calculated over commands or responses in a way that is specified below. The size is 4 bytes and these 4 bytes can hold any value. The TI is treated as a byte array, so there is no notion of MSB and LSB.

Both sides know the TI after being generated by the PICC on successful execution of the authentication protocol initiated by AuthenticateEV2First, as it is transmitted as part of the AuthenticateEV2FirstPart2 response from PICC to PCD so that both sides have the same value.

In case of ECC-based authentication, see Section 7.5.6, it is expected that a transaction will only consist of a single authentication. As a KeyID.CARootKey and/or reader certificate can cover multiple access rights, see Section 7.6, there should not be a need to authenticate multiple times. Therefore, in VCState.AuthenticatedECC, the TI is set to all zero bytes.

7.5.10.2 Command Counter

Messages will be partly encrypted and partly authenticated via a MAC as will be seen below. However it is also important to make sure that an attacker cannot delete messages from a transaction unnoticed, nor insert messages, even if they are replayed messages that are therefore authenticated, leading to the wrong results. In order to prevent this, a command counter is used.

The command counter is included in the MAC calculation for commands and responses. It is also used to construct the IV for encryption and decryption.

Each command (besides few exceptions see below) is counted by the command counter (called CmdCtr). Both sides count commands, so the actual value of the CmdCtr never needs to be transmitted.

CmdCtr is a 16-bit unsigned integer. The CmdCtr is reset to 0x0000 at PCD and PICC after a successful authentication resulting from the AuthenticateEV2First. The counter is maintained as long as the PICC remains in VCState.DFAuthenticatedEV2. In cryptographic calculations, the CmdCtr is represented LSB first.

Subsequent authentications using AuthenticateEV2NonFirst, so where the VC state is VCState.AuthenticatedAES and stays so, do not affect the CmdCtr (neither increasing, nor resetting). Subsequent authentications using the AuthenticateEV2First will reset CmdCtr to 0x0000.

The CmdCtr is increased between the command and response, for all communication modes. Note that for CommMode.Plain, this is not reflected in the actual command exchange as the CmdCtr is not used. For CommMode.MAC, the following holds. At PCD side, when a MAC on a command must be calculated that includes the CmdCtr it must use the current CmdCtr. The CmdCtr must be afterward increment by 1. At PICC side, the MAC of received commands which must be accompanied by a MAC including the CmdCtr, must be checked using the current value of CmdCtr. If the MAC matches, CmdCtr must be increment by 1 after successful reception of the command, but before sending a response. For CommMode.Full, the same holds

Provided under NDA Agreement - Personal copy for: COMPANY CONFIDENTIAL Quixtar Solutions GmbH Do not copy, share or reproduce 5fc07f7d-0224-455c-99fc-6ec3ca42

for both the MAC and encryption IV calculation, that is, the non-increased value is used for the command calculations while the increased value is used for the response calculations.

If the CmdCtr holds the value 0xFFFF and a command maintaining the active authentication arrives at the PICC, this must lead to an error being returned and further handling as described for the case where the MAC was wrong.

Command chaining, see Section 7.5.2, does not additionally affect the counter increase. The chained command is considered as a single command, just as for the other aspects of secure messaging, and therefore the related counter is increased only once.

The MIFARE Proximity commands, see Section 7.18 do not increase the command counter, as they are not protected by the EV2 Secure Messaging.

Note that a command using CmdCtr is handled as follows. If CmdCtr = *n* at the moment the application wants to send the command, the MAC on the command is calculated including CmdCtr = *n*. The MAC on the response will be calculated over a value including CmdCtr = *n + 1*.

7.5.10.3 MAC Calculation

MACs are calculated using the underlying block cipher according to the CMAC standard described in NIST Special Publication 800-38B [3]. Padding is applied according to the standard.

MF3E(H)x3 supports AES-128 and AES-256 as the underlying block cipher [1].

This document assumes familiarity with the details of the AES algorithm as specified in [1]. For clarity in what follows, here we only remind the reader of the byte numbering convention for the state array of an AES computation. The AES cipher operates on a state array of bytes, which is of size 4 bytes by 4 bytes. Table 13 below shows this state array with entries $S_{r,c}$ where r and c are the row and column indexes respectively.

Table 13. AES state array byte ordering

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

In [1], it is shown that when representing the outputs of an AES cipher operation as a bitstring, the bytes are ordered as follows:

$$S_{0,0} \parallel S_{1,0} \parallel S_{2,0} \parallel S_{3,0} \parallel S_{0,1} \parallel S_{1,1} \parallel S_{2,1} \parallel S_{3,1} \parallel S_{0,2} \parallel S_{1,2} \parallel S_{2,2} \parallel S_{3,2} \parallel S_{0,3} \parallel S_{1,3} \parallel S_{2,3} \parallel S_{3,3}$$

We subsequently choose to number the bytes of this bit string according to a convention in which S_{15} is the left most or most significant byte and S_0 is the rightmost or least significant byte, so that $S_{r,c} = S_{15 - (4c + r)}$.

Within [3], the standard method for truncation of a MAC is to retain the required number of most significant bits; accordingly, truncation of an AES-based MAC to 8 bytes corresponds to retention of final state bytes S_{15} to S_8 inclusive.

This MAC calculation adopts an alternative MAC truncation method. If $S_{15}..S_0$ denote the 16 bytes of a MAC computation according to [3] and ordered according to the numbering convention described above, then the MAC truncation function MACt() is defined to be:

$$MACt(key,message) = S_{14} \parallel S_{12} \parallel S_{10} \parallel S_8 \parallel S_6 \parallel S_4 \parallel S_2 \parallel S_0$$

that is, the even-numbered bytes are retained in most-to-least-significant order. *MAC(key,message)* denotes the CMAC result before truncation.

Provided under NDA only. COMPANY CONFIDENTIAL - Personal copy for: Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-405c-90fe-6ec3cae42f1

Initialization Vector for MACing

The initialization vector used for the CMAC computation is the zero byte IV as prescribed [3].

7.5.10.4 Encryption

Encryption and decryption are calculated using the underlying block cipher according to the CBC mode of NIST SP800-38A [2].

MF3E(H)x3 supports AES-128 and AES-256 as the underlying block cipher [1].

Padding is applied according to Padding Method 2 of ISO/IEC 9797-1, [12], that is, by adding always 0x80 followed, if required, by zero bytes until a string with a length of a multiple of 16 bytes is obtained. There are no exceptions. Note that if the plain data is a multiple of 16 bytes already, an additional padding block is added. The only exception is during the authentication itself (AuthenticateEV2First and AuthenticateEV2NonFirst), where no padding is applied at all.

The notation *E(key,message)* is used to denote the encryption and *D(key,message)* for decryption.

Initialization Vector for Encryption

When encryption is applied to the data sent between the PCD and the PICC, the Initialization Vector (IV) is constructed by encrypting with KeyID.SesAuthENCKey according to the EBC mode of NIST SP800-38A [2] the concatenation of:

- a 2-byte label, distinguishing the purpose of the IV: 0xA55A for commands and 0x5AA5 for responses
- TI
- CmdCtr (LSB first)
- zero padding

This results in the following IVs:

IV for CmdData = $E(K_{SesAuthENC}; 0xA5 \| 0x5A \| TI \| CmdCtr \| 0x0000000000000000)$

IV for RespData = $E(K_{SesAuthENC}; 0x5A \| 0xA5 \| TI \| CmdCtr \| 0x0000000000000000)$

When an encryption or decryption must be calculated, the CmdCtr to be used in the IV are the current values. Note that this means that if CmdCtr = *n* before the reception of a command, this value will be used if any encryption on the command. For the response, that is, after the validation of the command, the CmdCtr is incremented: CmdCtr = *n + 1*, and that value will be used in the IV for the encryption of the response.

For the encryption during authentication (both AES-based and ECC-based authentications), the IV will be 128 bits of 0.

7.5.10.5 Session Key Generation

As an output of a successful authentication, both the PICC and the PCD have generated two session keys for secure messaging:

- KeyID.SesAuthMACKey or $K_{SesAuthMAC}$ for MACing of messages
- KeyID.SesAuthENCKey or $K_{SesAuthENC}$ for encryption and decryption of messages

These session keys are generated differently, depending on the authentication:

- For ECC-based authentication, this is defined in Section 7.5.6.4.
- For AES-based authentication, this is defined in Section 7.5.8.4.

Provided under NDA only, CONFIDENTIAL - Personal copy for: Sergio Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc07f9d-c22c-4f3-90fe-6ecc3ce42f1

7.5.10.6 Plain Communication Mode

The command and response data is not secured. The data is sent in plain, that is, as defined in the command specification tables, see Command set.

However, note that, as the PICC is in state VCState.DFAuthenticatedEV2, the command counter CmdCtr is still increased as defined in Section 7.5.10.2.

This allows the PCD and PICC to detect any insertion and/or deletion of commands sent in CommMode.Plain on any subsequent command that is sent in CommMode.MAC (for example, CommitTransaction) or CommMode.Full.

7.5.10.7 MAC Communication Mode

The EV2 Secure Messaging applies CommMode.MAC to all commands listed as such in Table 8.

So, in case CommMode.MAC is to be applied, the following holds. The MAC is calculated using the current session key KeyID.SesAuthMACKey. MAC calculation is done as defined in Section 7.5.10.3.

For commands, the MAC is calculated over the following data in this order:

- Cmd
- CmdCtr
- TI
- CmdHeader (if present)
- CmdData (if present)

For responses, the MAC is calculated over the following data in this order:

- RC
- CmdCtr (The already increased value)
- TI
- RespData (if present)

CmdCtr is the Command Counter as defined in Section 7.5.10.2. Note that the CmdCtr is increased between the computation of the MAC on the command and the MAC on the response. TI is the Transaction Identifier, as defined in Section 7.5.10.1. The other input parameters are as defined in Section 7.5.1. The calculation is illustrated in Figure 7.

In case of command chaining, the MAC calculation is not interrupted. The MAC is calculated over the data including the complete data field (that is, either CmdData or RespData of all frames) at once. The MAC is always transmitted by appending to the un-padded plain command. If necessary, an additional frame is sent.

If a MAC over the command is received, the PICC verifies the MAC and rejects commands that do not contain a valid MAC by returning INTEGRITY_ERROR. In this case, the ongoing command and transaction are aborted (see also Section 7.5.3). The authentication state is immediately lost and the error return code is sent without a MAC appended. Note that any other error during the command execution has the same consequences.

Provided under NDA only
COMPANY CONFIDENTIAL - personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

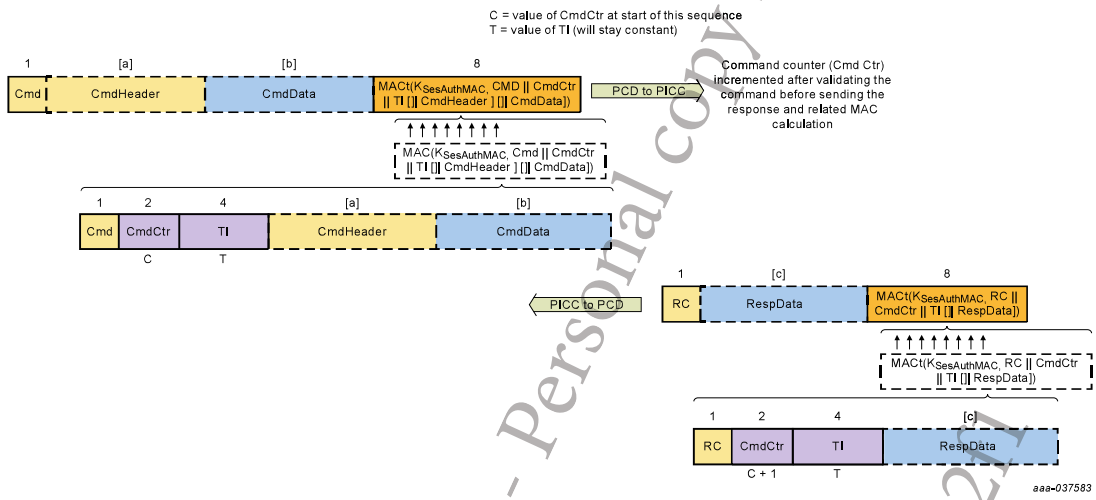


Figure 7. EV2 Secure Messaging: MAC protection mode

7.5.10.8 Full Communication Mode

The EV2 Secure Messaging applies CommMode.Full to all commands listed as such in Table 8.

So, in case CommMode.Full is to be applied, the following holds. The encryption/decryption is calculated using the current session key KeyID.SesAuthENCKey. Calculation is done as defined in Section 7.5.10.4 over either the command or the response data field (that is, CmdData or RespData). Note that none of the commands have a data field in both the command and the response frame.

After the encryption, the command and response frames are handled as with CommMode.MAC. This means additionally a MAC is calculated and appended for transmission using the current session key KeyID.SesAuthMACKey. This is exactly done as specified for CommMode.MAC in Section 7.5.10.7, replacing the plain CmdData or RespData by the encrypted field: $E(K_{SesAuthENC}, \text{CmdData})$ or $E(K_{SesAuthENC}, \text{RespData})$. The complete calculation is illustrated in Figure 8.

In case of command chaining, the encryption/decryption is applied over the complete data field (that is, of all frames). If necessary, due to the padding or the MAC added, an additional frame is sent.

If encryption of the command is required, after the MAC verification as described for CommMode.MAC, the PICC verifies and removes the padding bytes. Commands without a valid padding are also rejected by returning INTEGRITY_ERROR. In this case, the ongoing command and transaction are aborted (see also Section 7.5.3). The authentication state is immediately lost and the error return code is sent without a MAC appended. Note that any other error during the command execution has the same consequences.

Provided under NXP Company COMPANY CONFIDENTIAL - Personal copy for: Sergii Salatenko, Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-4b5c-9066-6e63cae421

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

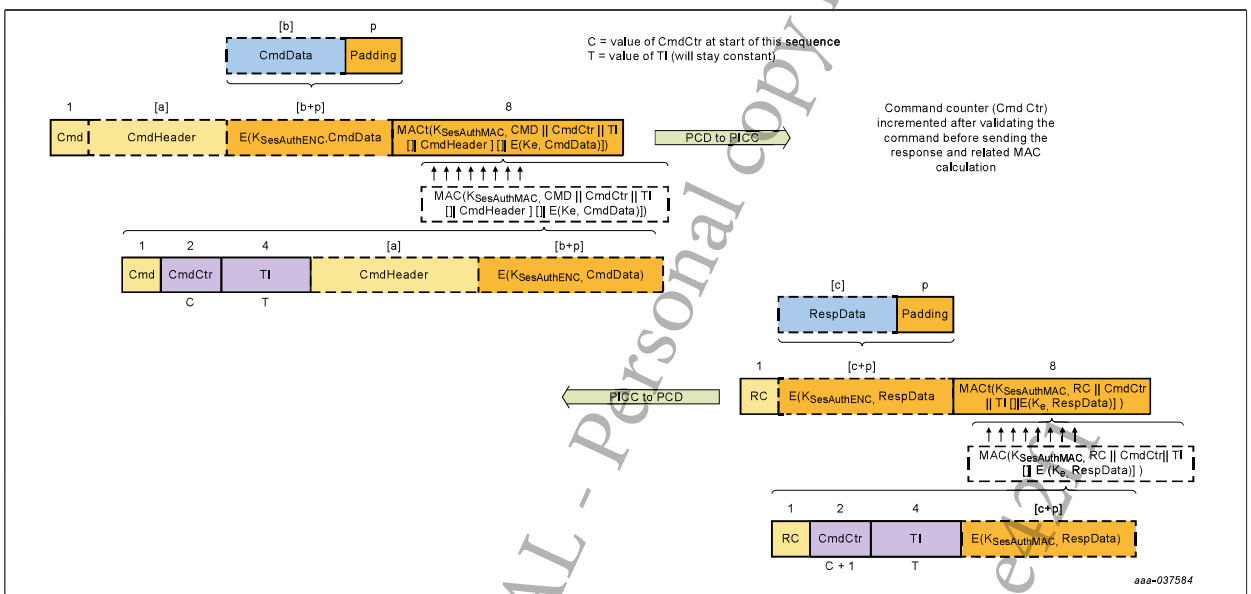


Figure 8. EV2 Secure Messaging: Full protection mode

7.5.11 Secure Dynamic Messaging

The Secure Dynamic Messaging (SDM) allows for confidential and integrity-protected data exchange, without requiring a preceding authentication.

MF3E(H)x3 supports SDM for reading from one or more of the FileType.StandardData files on the PICC (for file type definition, see Section 7.12). Secure Dynamic Messaging allows adding security to the data read, while still being able to access it with standard NDEF readers for NFC Forum Tag Type 4 formatted cards. The typical use case is an NDEF holding a URI and some meta-data, where SDM allows this meta data to be communicated confidentiality and integrity protected toward a backend server.

However, note also the additional residual risks coming with this feature, as it does not give the same aliveness guarantees as a challenge-response protocol, that is, anybody can read out the message and potentially play to the verifier at a later time. This residual risk needs to be assessed by the system use case.

MF3E(H)x3 supports two modes for integrity protection and authentication of the data:

- symmetric SDMMAC, where the data is protected by a Message Authentication Code which is generated by a symmetric AES key. This is specified in Section 7.5.11.1.8 and Section 7.5.11.1.9.
- asymmetric SDMSIG, where the data is protected by a Signature which is generated with the private key of an ECC key pair. This is specified in Section 7.5.11.1.10 and Section 7.5.11.1.11. This approach may ease the key management towards for example, reader infrastructure as no secret key is required for the signature validation.

Encryption is always based on symmetric cryptography, as specified in Section 7.5.11.1.4 for PICCData like the VCUID and Section 7.5.11.1.7 for generic file data (SDMENCFileData).

The session key derivation for symmetric keys (be it for encryption or MACing) is outlined in Section 7.5.11.1.12.

SDM is enabled and configured with ChangeFileSettings, see Section 7.12.2.3. Access right related aspects are defined in Section 7.12.2.1.

Printed under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Salazar Mendoza
Quality Management GmbH
Do not copy, share or reproduce
5fc07791-22d0-4056-9016-ecc3caef4221

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.5.11.1 SDM for Reading

7.5.11.1.1 SDM Read Counter

In order to allow replay detection by the party validating the data read, a read counter is associated with the file for which Secure Dynamic Messaging is enabled.

SDMReadCtr is a 24-bit unsigned integer. The SDMReadCtr is reset to 0x000000 when enabling SDM with ChangeFileSettings. In cryptographic calculations and represented with binary encoding on the external interface, the SDMReadCtr is represented LSB first. When represented with ASCII encoding on the external interface, it is represented MSB first for compatibility reasons with NTAG21x [3].

In VCState.NotAuthenticated, the SDMReadCtr is incremented by 1 before calculating the response of the first read command, ReadData or ISOReadBinary, if successful. On subsequent read commands targeting the same file the SDMReadCtr is not increased, and the current value is used. As soon as a different command has been received, the counter is incremented again on a subsequent read command. Also when varying between ReadData and ISOReadBinary, the counter is incremented on each first instance of read command type. This depends on the command code. When switching between ReadData 0xBD and 0xAD, the counter is incremented, but when switching between ISO/IEC 7816-4 wrapped and native command format with the same command code it is not.

Note that the SDMReadCtr is not incremented when authenticated, that is, when not in VCState.NotAuthenticated.

If the SDMReadCtr reaches the SDMReadCtrLimit (see next section) or the value 0xFFFFFF (if SDMReadCtrLimit is not enabled) and a first read command arrives at the PICC, this will lead to an error being returned. Command chaining, see Section 7.5.2, does not additionally affect the counter increase. The chained command is considered as a single command.

Note that if multiple files have been configured for SDM, each file has its own SDMReadCtr.

SDMReadCtr can be retrieved via the mirroring as part of the PICCData, see Section 7.5.11.1.3, or it can be retrieved via GetFileCounters.

7.5.11.1.2 SDM Read Counter Limit

In order to allow limiting the number of reads that can be done with a single device applying Secure Dynamic Messaging, an optional SDM Read Counter Limit can be configured.

The main use case is to limit the number of usages from the card side. Note that typically this can also be controlled from the backend verifying the SDM for Read protected message.

The number of reads that can be executed for an SDM configured file can be limited by setting an SDM Read Counter Limit (SDMReadCtrLimit). This is an unsigned integer of 3 bytes, related with SDMReadCtr. On the interface, the SDMReadCtrLimit is represented LSB first. The SDMReadCtrLimit can be enabled by setting a customized value with ChangeFileSettings. It can be retrieved with GetFileSettings.

Once the SDMReadCtr equals the SDMReadCtrLimit, no reading of the file with ReadData or ISOReadBinary can be executed, inVCState.NotAuthenticated. Note that if authenticated reading is always possible even if SDMReadCtrLimit is reached, applying the regular secure messaging.

If the SDMReadCtrLimit is disabled with ChangeFileSettings, this is also equivalent to putting it to the maximum value: 0xFFFFFF.

Note that enabling the SDMReadCtrLimit might create a DoS risk, as the file can be freely read, incrementing SDMReadCtr each time.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Santa
Quantum Solutions GmbH
Do not copy, share or reproduce
5fc07790-c2c4-4056-90fc-6ec3cae8a7

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.5.11.1.3 PICCData

The PICCData holds meta data of the targeted PICC and file, consisting of the VCUID and/or SDMReadCtr.

Whether PICCData is plain or encrypted depends on the configuration of FileAR.SDMMetaRead.

If FileAR.SDMMetaRead is configured for free access (0xE), PICCData will be plain and is defined according to Table 14.

ASCII mirroring is reflected by the function EncodeASCII(), which means that each hexadecimal character of the hexadecimal representation will be ASCII encoded using capitals. For example, the VCUID 0x04E141124C2880 becomes: 0x30 0x34 0x45 0x31 0x34 0x31 0x31 0x32 0x34 0x43 0x32 0x38 0x38 0x30.

Table 14. PICCData: plain encoding and lengths

Mode	PICCData Value	Length
ASCII	EncodeASCII(VCUID)	VCUIDLength = 14 or 20 (that is, 2*UIDLen)
ASCII	EncodeASCII(SDMReadCtr)	SDMReadCtrLength = 6 (that is, 2*3)

Note that the SDMReadCtr, as defined in Section 7.5.11.1.1, is represented MSB first for ASCII case.

If FileAR.SDMMetaRead is configured for an application key, PICCData will be encrypted as defined in Section 7.5.11.1.4. In this case, the input plaintext for the encryption is always in binary encoding, while the output ciphertext will be ASCII encoded.

The PICCData is mirrored within the file. This is configured with ChangeFileSettings, see Section 7.12.2.3 via the related offsets.

In case of plain mirroring (that is, FileAR.SDMMetaRead = 0xE):

- VCUIDOffset configures the VCUID mirroring position. It is only given if VCUID mirroring is enabled. Note that if the card is configured for Random ID, the actual VCUID shall not be mirrored in plain, but will be replaced by a string of zero bytes of the same length. In this case, encrypted mirroring should be used. This behavior was added to MF3E(H)x3 compared to NTAG42x to avoid the “accidental” breaking of untraceability via this feature especially in the context of delegated application.
- SDMReadCtrOffset configures the SDMReadCtr mirroring position. It is only given if SDMReadCtr mirroring is enabled. Note that it is possible to enable the SDMReadCtr but without mirroring by putting SDMReadCtrOffset to 0xFFFFFF. In this case, it can be retrieved with GetFileCounters.

If VCUID and SDMReadCtr are mirrored within the file, they shall not overlap:

- VCUIDOffset ≥ SDMReadCtrOffset + SDMReadCtrLength OR SDMReadCtrOffset ≥ VCUIDOffset + VCUIDLength.

In case of encrypted mirroring (that is, FileAR.SDMMetaRead = 0x0..0xD), PICCDataOffset configures the PICCData mirroring. The encryption is outlined in Section 7.5.11.1.4.

If the PICCData is mirrored within the file, the mirroring shall always be applied in VCState.NotAuthenticated, independently of whether Secure Dynamic Messaging applies. This means it will also be applied if reading the file with free access due to FileAR.Read or FileAR.ReadWrite. Note that if authenticated, that is, in VCState.AuthenticatedAES, no mirroring is done, that is, the regular secure messaging is always applied on the static file data.

With MF3E(H)x3, PICCData will always be ASCII encoded.

When both the VCUID and SDMReadCtr are mirrored, “x” (0x78) is used as a separator character with NTAG2x. This can be achieved by leaving one-byte space between the placeholders defined by VCUIDOffset and SDMReadCtrOffset, and writing “x” (0x78) in the static file data.

Provided under NDA only. CONFIDENTIAL - Personal copy for: Original Data Solutions GmbH Contact IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-f206-4080-8ecc3ca42f1

7.5.11.1.4 Encryption of PICCData

In case of encrypted PICCData mirroring (both binary and ASCII), PICCDataTag specifies what metadata is mirrored, together with the length VCUID if mirrored, as defined in Table 15.

The format of the plain text is: PICCDataTag [|| VCUID] [|| SDMReadCtr] || RandomPadding.

To ensure that the encrypted PICCData cannot be abused for tracking purposes, random padding is added to the actual input. Note that the random padding is always added, but to avoid the tracking this is only relevant if the SDMReadCtr is not mirrored, as in that case the input is already always different.

Note that in case of 10-byte VCUID, if the SDMReadCtr is not mirrored, the random padding is added immediately after the PICCDataTag, and padding extends the output to two blocks. The format of the plain text is then: PICCDataTag || RandomPadding || VCUID. with 21 bytes of RandomPadding.

The random padding is generated for the response of the first read command, ReadData or ISOReadBinary. On subsequent read commands targeting the same file the same random padding is reused. This allows for reading the file in chunks, where a chunk border might even be in the middle of the encrypted PICCData. As soon as a different command has been received, fresh random padding is generated on a subsequent read command. Also when varying between ReadData and ISOReadBinary, fresh random padding is generated.

The key applied for encryption of PICCData is the KeyID.SDMMetaReadKey as defined by FileAR.SDMMetaRead.

Table 15. PICCDataTag

Bit	Value	Description
Bit 7		VCUID mirroring
	0	disabled
	1	enabled
Bit 6		SDMReadCtr mirroring
	0	disabled
	1	enabled
Bit 5-4	00	RFU
Bit 3-0		VCUID Length
	0x0	RFU (if VCUID is not mirrored)
	0x7	7 byte VCUID
	0xA	10 byte VCUID

Encryption and decryption of the PICCData are calculated using the underlying block cipher according to the CBC mode of NIST SP800-38A [2], applying zero byte IV. Therefore PICCData is defined as follows:

$$PICCData = E(K_{SDMMetaRead}, PICCDataTag[|| VCUID] [|| SDMReadCtr] || RandomPadding)$$

with PICCDataTag as defined in PICCData, and RandomPadding being a random byte string generated by the PICC to make the input 16 bytes long. There is one exception: in case of 10-byte UID and not mirroring the SDMReadCtr, random padding up to 32 bytes will be added immediately after the PICCDataTag:

$$PICCData = E(K_{SDMMetaRead}, PICCDataTag || RandomPadding || VCUID)$$

Note that in case of ASCII encoding the required placeholder length is 32 bytes (or 64 bytes in case of 10-byte UID and not mirroring the SDMReadCtr).

Provided under NDA only
CONFIDENTIAL Property of:
COMPANY CONFIDENTIAL
Sergi Sata
Qualitas Solutions GmbH
Do not copy, share or reproduce
5fc07796-c22-405c-90fe-6ecc3ca2a99

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.5.11.1.5 GPIOStatus

When one of the GPIO pins is configured for input, see Section 7.16, with SetConfiguration 0x11, see Section 7.7.3.1, it is possible to mirror the statuses within the NDEF file.

The GPIO statuses are encoded on a 3-byte string identical to the ReadGPIO response, see Section 7.16.3 and especially Table 47.

They can be mirrored in plain or encrypted. For the latter, GPIOStatus needs to be positioned within the placeholder for the plain data that serves as input for SDMENCFileData, see next section. In this case, the static file data is replaced by the dynamic statuses before applying the encryption. Note, however, that either all status bytes are plain, or all are encrypted.

As the status bytes are already ASCII encoded, no ASCII encoding needs to be applied on top, and only a 3-byte placeholder is required. Where the status is mirrored within the file, is configured with ChangeFileSettings, see Section 7.12.2.3 via GPIOStatus. The restrictions on this offset shall be that it may not be overlapping with any PICCData mirrored or with the SDMMAC.

If the GPIOStatus is mirrored within the file, the mirroring shall always be applied in VCState.NotAuthenticated, independently of whether Secure Dynamic Messaging applies. This means it will also be applied if reading the file with free access due to FileAR.Read or FileAR.ReadWrite. Note that if authenticated, that is, in VCState.AuthenticatedAES or VCState.AuthenticatedECC, no mirroring is done, that is, the regular secure messaging is always applied on the static file data.

7.5.11.1.6 SDMENCFileData

SDM for Reading supports mirroring (part of the) file data encrypted. This part is called the SDMENCFileData. If FileAR.SDMFileRead is configured for an application key, part of the file data can optionally be encrypted as defined in Section 7.5.11.1.7 when being read out in VCState.NotAuthenticated.

In this case, the input plaintext for the encryption is always in binary encoding, while the output ciphertext will be ASCII encoded, according to the configuration.

Note that if authenticated no Secure Dynamic Messaging is applied, that is, the regular secure messaging is always applied on the static file data.

The SDMENCFileData (if any) is always mirrored within the file. This is configured with ChangeFileSettings, see Section 7.12.2.3 via SDMENCOffset and SDMENCLength.

If FileAR.SDMFileRead is disabling Secure Dynamic Messaging for reading (that is, set to 0xF), SDMENCOffset and SDMENCLength are not present in ChangeFileSettings.

If PICCData is mirrored within the file, SDMENCFileData shall not overlap with it. Depending on what is exactly mirrored, the following holds:

- SDMENCOffset ≥ PICCDataOffset + PICCDataLength OR PICCDataOffset ≥ SDMENCOffset + SDMENCLength.
- SDMENCOffset ≥ VCUIDOffset + VCUIDLength OR VCUIDOffset ≥ SDMENCOffset + SDMENCLength.
- SDMENCOffset ≥ SDMReadCtrOffset + SDMReadCtrLength OR SDMReadCtrOffset ≥ SDMENCOffset + SDMENCLength.

It shall be ensured that SDMENCOffset + SDMENCLength is smaller than or equal to the file size. Note that if the SDMMAC is mirrored as well within the file, additional conditions apply, see next section.

The SDMENCLength shall be a multiple of 32 bytes for ASCII encoding. With MF3E(H)x3, only ASCII encoding is supported.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-022c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.5.11.1.7 Encryption of SDMENCFileData

The key applied for the encryption is a session key KeyID.SesSDMFileReadENCKey derived from the application key defined by FileAR.SDMFileRead, as specified in Section 7.5.11.1.12.

From user point of view, the SDMENCOffset and SDMENCLength define a placeholder within the file where the plain data is to be stored when writing the file.

For ASCII encoding, only the first half of the placeholder is used for storing the plain data, the second half will be ignored for constructing the returned data when reading with SDM. For example, if targeting to encrypt 2 AES blocks, that is, 32 bytes, a placeholder of 64 bytes is reserved via SDMENCOffset and SDMENCLength. The first 32 bytes will hold the plaintext, and the next 32 bytes are ignored when reading with Secure Dynamic Messaging.

Encryption and decryption of the SDMENCFileData are calculated using the underlying block cipher according to the CBC mode of NIST SP800-38A [17]. The following IV is applied:

$$IV = E(K_{SesSDMFileReadENC}, SDMReadCtr \,||\, 0x00000000000000000000000000)$$

with SDMReadCtr LSB first.

For applying SDM with ASCII encoding, the SDMENCFileData is defined as follows:

$$SDMENCFileData = E(K_{SesSDMFileReadENC}, StaticFileData[SDMENCOffset.. SDMENCOffset + SDMENCLength=2 - 1])$$

with StaticFileData being the current file data as written in the placeholder. The file configuration ensures via SDMENCLength that the input is a multiple of 16 bytes, so no padding is applied.

Note that it is possible via the read command parameters to read-only part of the file. If the SDMENCFileData is partially read as per the issued offset and length, a truncated part of the ciphertext will be returned. As truncation might happen in the middle of an AES block, this means subsequent read commands to fetch the remainder of the file might be required to be able to decrypt.

Note that the SDMENCFileData is mirrored inside the file and does not depend on the actual read command sent.

7.5.11.1.8 SDMMAC

SDM for Reading supports calculating a MAC over the response data. This message authentication code is called the SDMMAC.

If FileAR.SDMFileRead is configured for an application key, a MAC will be calculated as defined in Section 7.5.11.1.9 when being read out in VCState.DFNotAuthenticated.

The SDMMAC is to be mirrored within the file via SDMMACOffset. This is configured with ChangeFileSettings, see Section 7.12.2.3.

If SDMMAC is mirrored within the file, it shall be limited to only start after SDMENCFileData, that is, SDMMACOffset ≥ SDMENCOffset + SDMENCLength. The SDMMACInputOffset must ensure that the complete SDMENCFileData is included in the MAC calculation.

In case of ASCII encoding, the output size doubles to 16 bytes.

It shall be ensured that SDMMACOffset + SDMMACLength is smaller or equal than the file size. Note that if authenticated no Secure Dynamic Messaging is applied and the placeholder data at SDMMACOffset is not replaced, that is, the regular secure messaging is always applied on the static file data.

The SDMMACInputOffset will define from which position in the file the MAC calculation starts. If SDMMAC is mirrored within the file, SDMMACInputOffset shall be smaller than or equal to SDMMACOffset.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Scandata Quality Solutions GmbH
Do not copy, share or reproduce
5fc07f7d-c2c4-4056-901e-6cc3cae42f1

MACing is mandatory if FileAR.SDMFileRead is configured for an application key. If FileAR.SDMFileRead is disabling Secure Dynamic Messaging for reading (that is, set to 0xF), SDMMACOffset and SDMMACInputOffset are not present in ChangeFileSettings.

With MF3E(H)x3, only ASCII encoding is supported. SDMMAC shall always be mirrored within the file.

7.5.11.1.9 MAC Calculation

The key applied for the MAC calculation is a session key KeyID.SesSDMFileReadMACKey derived from the application key defined by FileAR.SDMFileRead, as specified in Section 7.5.11.1.12.

The 8-byte SDMMAC is calculated using the underlying block cipher according to the CMAC standard described in NIST Special Publication 800-38B [18] applying the same truncation as the EV2 secure messaging, see Section 7.5.10.3.

If the SDMMAC is to be mirrored within the file, it is defined as follows:

$$SDMMAC = MACt(K_{SesSDMFileReadMAC}, DynamicFileData[SDMMACInputOffset.. SDMMACOffset - 1])$$

with *DynamicFileData* being the file data as how it is put on the external interface, that is, replacing any placeholders by the dynamic data.

7.5.11.1.10 SDMSIG

If FileAR.SDMFileRead2 is configured for an application KeyID.ECCPrivateKey, a signature, called SDMSIG, will be calculated as defined in Section 7.5.11.1.11 when being read out in VCState.NotAuthenticated. If the targeted key does not exist or is not enabled for ECC-based Secure Dynamic Messaging, no signature will be calculated and mirrored, and the static file data is returned. If the targeted KeyID.ECCPrivateKey has an enabled KeyUsageCtrLimit that was already reached, see Section 7.9.1.2, the read command is rejected.

The offsets for signature input and signature mirroring are configured with ChangeFileSettings, see Section 7.12.2.3. As to a large extent the same rules apply, parameters SDMMACOffset and SDMMACInputOffset are reused.

This means the SDMSIG is to be mirrored within the file via SDMMACOffset. It shall be limited to only start after SDMENCFileData, that is,SDMMACOffset ≥ SDMENCOffset + SDMENCLength. The SDMMACInputOffset must ensure that the complete SDMENCFileData is included in the signature calculation. The SDMSIGLength is 128 bytes, as only ASCII encoding is supported. It shall be ensured that SDMMACOffset +SDMSIGLength is smaller or equal than the file size.

Also here, if authenticated, no Secure Dynamic Messaging is applied and the placeholder data at SDMMACOffset is not replaced, that is, the regular secure messaging is always applied on the static file data.

The SDMMACInputOffset will define from which position in the file the input for the signature calculation starts. With MF3E(H)x3, only ASCII encoding is supported. SDMSIG shall always be mirrored within the file.

7.5.11.1.11 Signature Calculation

The key applied for the signature calculation is KeyID.ECCPrivateKey defined FileAR.SDMFileRead2.

The SDMSIG is calculated using ECDSA Digital Signature Generation as defined in [28]. The hash function to be applied is SHA-256, as specified in NIST FIPS 180-4 [25].

SDMSIG is defined as follows:

$$SDMSIG = ECDSA_{Sign}(Priv.x, DynamicFileData[SDMMACInputOffset.. SDMMACOffset - 1])$$

with `DynamicFileData` being the file data as how it is put on the external interface, that is, replacing any placeholders by the dynamic data.

Provided under NDA only
CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantum ITT Solutions GmbH
Do not copy, share or reproduce
5fc07799-c22c-45c-90fe-63a2cae4261

Note that the SDMSIG is mirrored inside the file and does not depend on the actual read command parameters (offset, length) and what part of the file is read. This means the SDMSIG may be read in chunks by several consecutive read commands with different offsets. It shall be ensured that these chunks when reassembled reflect a single, consistent signature.

7.5.11.1.12 Session Key Generation

For Secure Dynamic Messaging for reading, the following session keys are calculated:

- KeyID.SesSDMFileReadMACKey or $K_{SesSDMFileReadMAC}$ for MACing of file data.
- KeyID.SesSDMFileReadENCKey or $K_{SesSDMFileReadENC}$ for encryption of file data.

The session key generation is according to NIST SP 800-108 [19] in counter mode.

The pseudorandom function applied during the key generation is the CMAC algorithm described in NIST Special Publication 800-38B [3]. The key derivation key is the KeyID.SDMFileReadKey as configured with FileAR.SDMFileRead.

If the FileAR.SDMFileRead targets a KeyType.AES128 (see Section 7.8.2) key, the generated session keys are also of KeyType.AES128. If a KeyType.AES256 key is targeted, the session keys are also KeyType.AES256.

The input data is constructed using the following fields as defined by [19]. Note that NIST SP 800-108 allows defining a different order than proposed by the standard as long as it is unambiguously defined.

- a2-byte label, distinguishing the purpose of the key: 0x3CC3 for MACing and 0xC33C for encryption.
- a2-byte counter:
  - KeyType.AES128: fixed to 0x0001.
  - KeyType.AES256: counting from 0x0001 to 0x0002.
- a2-byte length:
  - KeyType.AES128: fixed to 0x0080.
  - KeyType.AES256: fixed to 0x0100.
- a context, constructed using the VCUID and/or SDMReadCtr, if needed, followed by zero-byte padding.

Whether the VCUID and/or SDMReadCtr are included in the session vectors, depends on whether they are mirrored, see Section 7.5.11.1.3.

Note: *If encrypting file data, mirroring of both is mandatory. Therefore, they are always included in SVx.*

KeyType.AES128

First, the input session vectors *SVx* are derived as follows:

SV1 = 0xC3 || 0x3C || 0x00 || 0x01 || 0x00 || 0x80 || VCUID || SDMReadCtr[|| ZeroPadding]
SV2 = 0x3C || 0xC3 || 0x00 || 0x01 || 0x00 || 0x80[|| VCUID][|| SDMReadCtr][|| ZeroPadding]

Padding with zeros is done up to a multiple of 16 bytes. So, for a 7-byte UID, no padding is added.

Then, the 16-byte session keys are constructed as follows:

KSesSDMFileReadENC= MAC(KSDMFileRead, SV1)
KSesSDMFileReadMAC=MAC(KSDMFileRead, SV2)

KeyType.AES256

First, the input session vectors *SVxx* are derived as follows:

SV1a=0xC3 || 0x3C || 0x00 || 0x01 || 0x01 || 0x00 || VCUID || SDMReadCtr[|| ZeroPadding]
SV1b=0xC3 || 0x3C || 0x00 || 0x02 || 0x01 || 0x00 || VCUID || SDMReadCtr[|| ZeroPadding]
SV2a=0x3C || 0xC3 || 0x00 || 0x01 || 0x01 || 0x00[|| VCUID][|| SDMReadCtr][|| ZeroPadding]
SV2b=0x3C || 0xC3 || 0x00 || 0x02 || 0x01 || 0x00[|| VCUID][|| SDMReadCtr][|| ZeroPadding]

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Sala
Quantum IT Solutions GmbH
Do not copy, share or reproduce
5107/9d-c22c-405c-99fe-6ecc3ca4ffa

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Padding with zeros are done up to a multiple of 16 bytes. So, for a 7-byte UID, no padding is added.

Then, the 32-byte session keys are constructed as follows:

KSesSDMFileReadENC= MAC(KSDMFileRead, SV1a) || MAC(KSDMFileRead, SV1b)
KSesSDMFileReadMAC= MAC(KSDMFileRead, SV2a) || MAC(KSDMFileRead, SV2b)

7.5.11.1.13 Output Mapping Examples

The following figure shows an example with the static file content and how it is read.

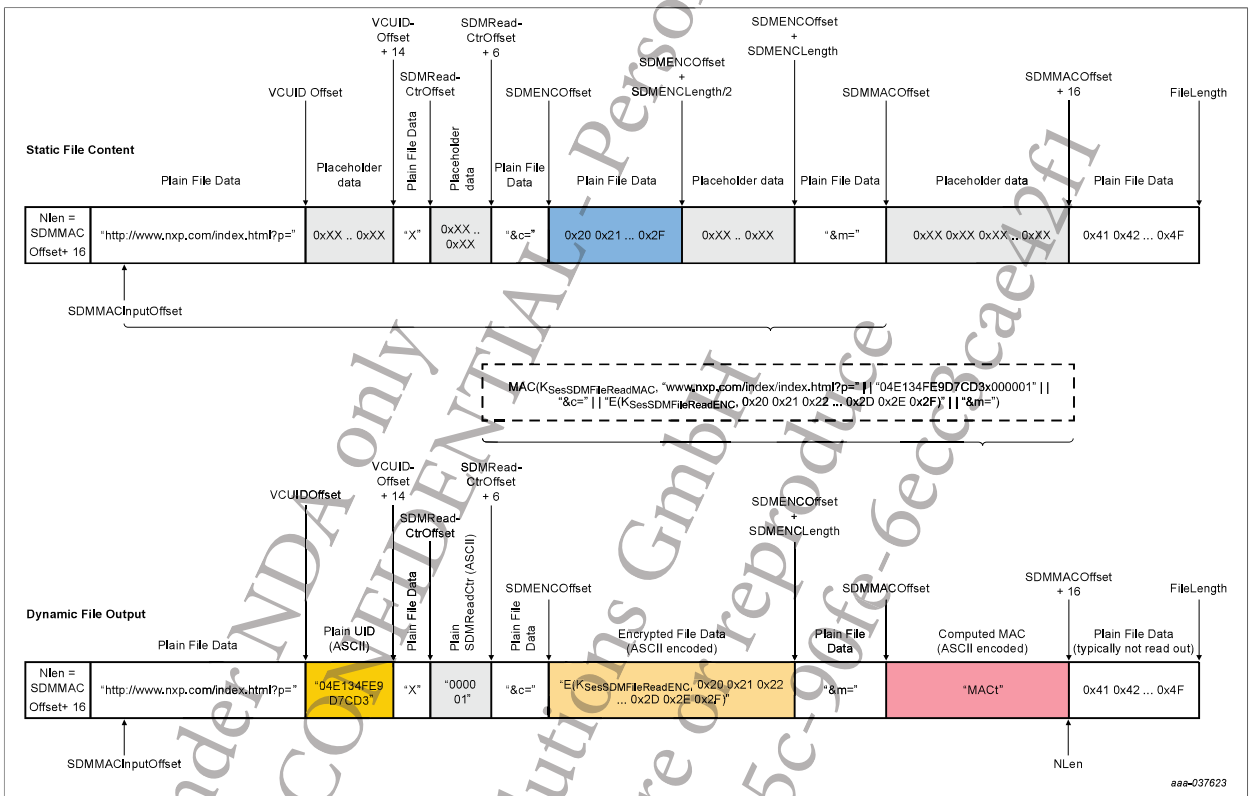


Figure 9. Secure Dynamic Messaging for Reading example

7.5.11.1.14 Transaction MAC Handling

In case a Transaction MAC calculation is ongoing when applying SDM when reading from a file, for the mirrored and SDM protected fields that are configured within the file, the dynamic SDM data as being exchanged over the interface will be included in the MAC calculation.

7.6 Access Rights Management

MF3E(H)x3 manages its access rights through access conditions. This is explained in Section 7.6.1. How access rights can be granted through certificates presented during asymmetric authentication is explained in Section 7.6.4.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sesa Sarta
Quantum Solutions GmbH
Do not copy, share or reproduce
5fc077d-c2ac-405c-96fe-6ec8cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.6.1 Access conditions

For file access, the conditions for the file access rights are associated with the file, as explained in Section 7.12.2. For other commands, the access conditions are either fixed or configurable via other means.

Nevertheless, the interpretation of access conditions and their representation in the command API is always the same. There are three kinds of access conditions:

- The *authentication* access conditions where a valid authentication is required. The access condition is satisfied by one of the following means:
  - An active symmetric authentication with the AuthKey (see Section 7.5.4) addressed by the key number encoded by the access condition.
    Note: *At PICC level, the access condition is not directly encoding the related (symmetric key number). PICC level access rights are further detailed in Section 7.6.2.*
  - An active asymmetric authentication granting the access condition via the current CertAccessRights. This authentication means the KeyID.CARootKey addressed during the authentication must have been associated with access rights encoded by the access condition. How a KeyID.CARootKey is configured with its access rights is defined in Section 7.9.2.
    Note: *Optionally, an individual reader certificate (or certificate chain) presented during the authentication can further restrict the granted access rights from the KeyID.CARootKey. This feature is specified in Section 7.6.4.*
- The *free access over I^2C* condition meaning the related commands can be accessed without an active authentication over the I^2C interface.
- The *free access* condition meaning the related commands can be accessed without an active authentication over any interface.
- The *no access* condition meaning no access to the related commands.

In other parts of the document, when it is stated that an active authentication with KeyID.AppKey is required, this means either a symmetric authentication with that particular key, or an asymmetric authentication granting equivalent access rights (even if the latter is not explicitly mentioned).

The access conditions are specified on 4 bits as defined in Table 16.

Table 16. Access condition values coded on 4 bits

Condition value	Description
0x0..0xD	authentication required
0xD	[Optional] free access over I^2C, authentication required over NFC
0xE	free access
0xF	no access or RFU

The access condition *free access over I^2C* needs to be explicitly enabled for an application through SetConfiguration Option 0x04 Bit 5. If this feature is enabled, it means a 0xD access condition can also still be obtained over the NFC interface, if authenticated with symmetric key 0xD, or obtaining 0xD access rights from an ECC-based authentication.

An access condition can be associated with any key number (0x0 . . . 0xD), even if those do not exist as a symmetric key within the targeted application. The latter can still be obtained via an asymmetric authentication. Concretely, this means that for an application with 5 KeyID.AppKeys, the access conditions 0x0..0x4 can be obtained both through symmetric and asymmetric authentication. The access conditions 0x5..0xD can be obtained through asymmetric authentication independently of the interface.

This is also illustrated by Figure 10 where access condition 0x0 can be obtained via a symmetric authentication targeting KeyID.AppKey 0x00, that is, the KeyID.AppMasterKey, but also through an asymmetric authentication targeting KeyID.CARootKey.1. This means that the latter also grants KeyID.AppMasterKey access rights.

Provided under IDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sarci Sabata
Quantum Solutions GmbH
Do not copy, share or reproduce
5f67ed c2-4c-405c-90fe-fecc3ca42f

In the case of asymmetric authentication, the access rights granted depend on the targeted KeyID.CARootKey, but can be further restricted via the certificate. If a certificate does not hold explicit access rights, the access rights from the related KeyID.CARootKey are implicitly inherited and therefore granted. When authenticating KeyID.CARootKey.1 of the example below, by default the accumulated access rights equivalent to being authenticated with either symmetric key 0x0 or key 0x5 is granted. However, Cert.ReaderB handed out by the CA related with KeyID.CARootKey.1, only grants access right 0x5, and therefore in that case not the KeyID.AppMasterKey access rights. Note that KeyID.CARootKey.1 only has two bits set, but there is no limitation on the number of bits and therefore access rights that can be granted to a KeyID.CARootKey: it could have all 14 bits set, granting access rights equivalent to a symmetric key 0x0 until 0xD.

Another example: if successfully authenticated with Cert.ReaderC, the user is granted access right 0x6 associated with KeyID.CARootKey.2. A CA shall not hand out certificates with access rights that exceed the access rights associated with KeyID.CARootKey. For example, a certificate with access right 0x3 handed out by the CA associated with KeyID.CARootKey.1 will not be accepted by MF3E(H)x3 as this access right is not configured for KeyID.CARootKey.1.

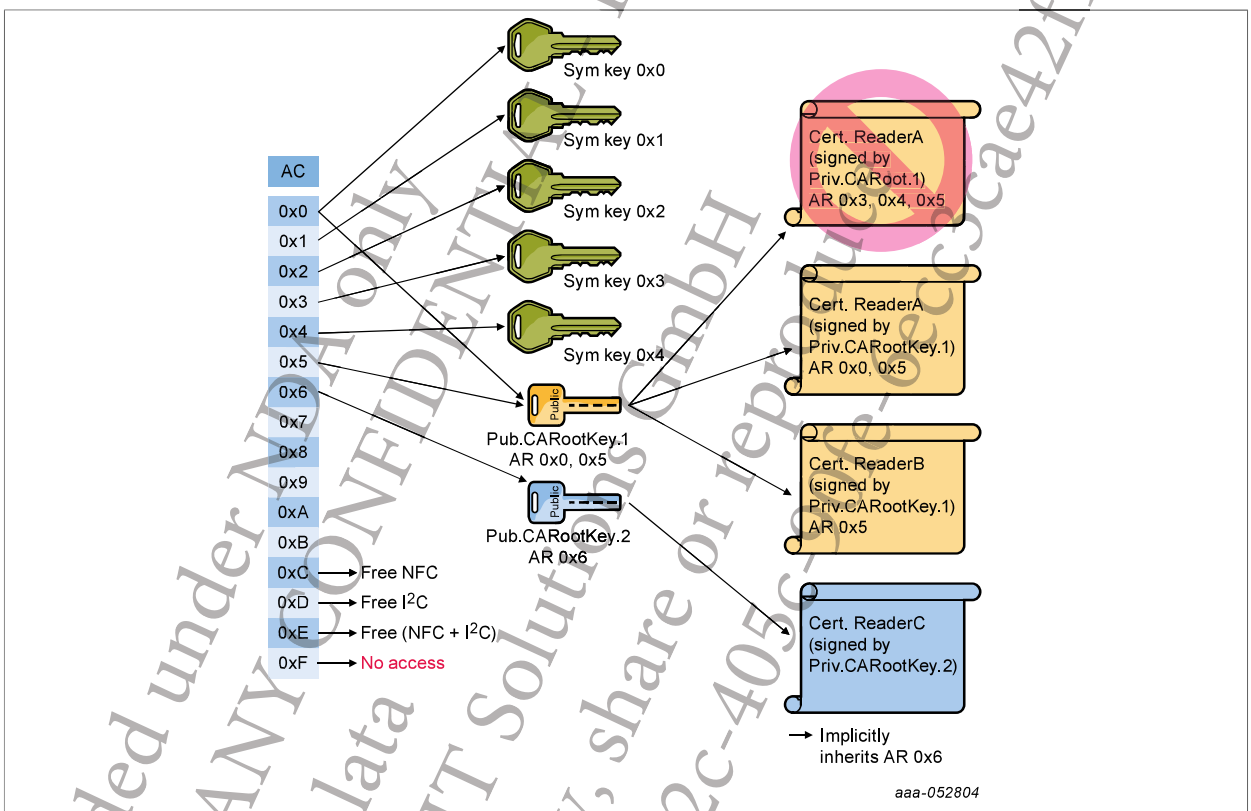


Figure 10. Access Conditions Example

7.6.2 PICC-level access rights

While at application level the access condition value directly maps to the key number of symmetric KeyID.AppKeys, as explained in Figure 10, this is not the case at PICC level. For the PICC level (AID 0x000000), the mapping between access conditions, symmetric keys, and the related granted access rights is defined in Table 16.

Provided under NDA only - Personal copy for: Sergio Salata - Quantag IT Solutions GmbH
CONFIDENTIAL - Do not copy, share or reproduce
5fc07f2d-c22c-405c-9fe-eecc5cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 17. Access condition (coded on 4 bits) mapping to access rights at PICC level

Condition value	Access right mapping
0x0	KeyID.PICCMasterKey
0x1	KeyID.VCConfigurationKey
0x2	ECC-based Delegated Application Management
0x3..0xD	ECC-specific access rights
0xE	free access
0xF	no access or RFU

The access right mappings mean that to fulfill access condition 0x0, either a symmetric authentication with KeyID.PICCMasterKey (KeyNo 0x00) is required, or an equivalent asymmetric authentication granting the access condition via the current CertAccessRights. In the latter case, also all access rights equivalent to symmetric KeyID.PICCMasterKey authentication are obtained, for example, to execute SetConfiguration functionality.

Similarly, for access condition 0x01, either a symmetric authentication with KeyID.VCConfigurationKey (KeyNo 0x20) is required, or an equivalent asymmetric authentication granting the access condition via the current CertAccessRights. Also here the latter allows to obtain equivalent access rights as when authenticated with KeyID.VCConfigurationKey.

Access condition 0x2 has a specific meaning. If this access condition is granted to the access rights associated with a KeyID.CARootKey at PICC level, certificates issued by this CA may allow for ECC-based Delegated Application Management, see Section 7.6.4.2. Note however that this access condition can as well be used as a regular access condition for for example, file access at the PICC-level.

AtPICC level, free access over I^2C is not supported.

Access conditions 0x3 until 0xD may be used by the user to restrict user configurable access, for example, WriteAccess of KeyID.ECCPrivateKey at PICC level, to ECC-based authentication, as those access conditions are not associated with symmetric PICC-level keys, or other additional functionality. Using access conditions in the upper half 0x8..0xD is recommended, because NXP may assign specific rights to access conditions 0x3 and higher in future evolutions. 0x3 is already used as the default for FileAR.Read of FileNo 0x00 storing the Cert.PICC at PICC level.

7.6.3 KeyID.CARootKey access rights

Access rights are associated with a KeyID.CARootKey through ManageCARootKey. For this command, the access conditions that can be granted when authenticated with this KeyID.CARootKey are encoded on a bitmap, as defined in Table 18. Note that, as defined in Section 7.5.1.1, this bitmap is sent on the interface LSB first.

Table 18. ACMap encoding

Bit Index	Description
Bit 15-14	RFU
Bit 13-0	ACbitmap. If bit 0 is set, AC 0x0 access rights granted. If bit 1 is set, AC 0x1 access rights are granted. And so on.

For these access rights the whole range of the AC bitmap can be used, independently of whether those bits encode key numbers of keys that exist in the targeted application, or at PICC level. For example, if the application holds five symmetric keys, the key numbers 0x0-0x4 (that is, bit 0 to bit 4 in the certificate encoding) can be used to specify access conditions that can be obtained by both symmetric and asymmetric

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salatenko, Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07794-992c-4905c-9004-6ec0cae42f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

authentication. From bit 5 onwards, the bits can only be used to specify access conditions that can be obtained via an asymmetric authentication. There does not exist an equivalent symmetric key within the application.

At PICC level, the same principle applies. However, at PICC-level the mapping to access rights as defined in Section 7.6.2 also applies for this encoding.

Access rights obtained during a mutual authentication can be further restricted via the presented certificates, as defined in Section 7.6.4. There the same encoding is used.

7.6.4 Certificate access rights

Overall certificate aspects are covered in Section 7.10. This includes the reader certificate definition, see Section 7.10.2, where the below described access right encoding is to be integrated in. Also certificate revocation is covered in Section 7.10.2.3.

MF3E(H)x3 supports a private extension (ARExtension) for access right encoding within X.509 certificates. This extension is reflected by an OID in the NXP range: 1.3.6.1.4.1.28137.64.1. If not present, the CertAccessRights are inherited from the parent certificate, or if there is no parent, the targeted KeyID.CARootKey. If multiple applications are selected, the access rights of the primary application are inherited from the KeyID.CARootKey targeted in the primary application; the access rights of the secondary application are inherited for the secondary KeyID.CARootKey targeted, that is, in the secondary application.

This ARExtension has the following ASN.1 encoding:

```
ARExtension  - SEQUENCE {
  arExtnId      OBJECT IDENTIFIER (id-nxp-ar),
  critical      BOOLEAN (TRUE),
  arExtnValue   OCTET STRING
}
```

CertAccessRights are obtained from a successful ECC-based Mutual or Reader-Unilateral Authentication, see Section 7.5.7. They are maintained as long as in VCState.AuthenticatedECC (see Section 7.5.4.

The access right extension value (ARExtensionValue) shall hold the data structure as defined in Table 19. The actual length and value format depend on the ARType. The total length is also encoded in the OCTET STRING encoding of the extension.

Table 19. CertificateAccess Right encoding

Field	Length/Bit Index	Description
ARType	1	Tag specifying the type of ARG
	Bit 7	CA delegation
		’0’: disabled (leaf)
		’1’: enabled (parent or leaf)
	Bit 6-0	ARType
		0x01: Application access rights, specified via AID
		0x02: Application access rights, specified via DFName
		0x03: Delegated Application Management access rights
ARValue	Variable	Actual access rights

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-9ffe-6ecc3caeb

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Bit 7 of the ARType indicates whether the certificate can be used as a parent certificate delegating access rights. Only if the bit is set, the certificate can be used to compose a certificate chain, representing an intermediate CA. In this case the certificate can also still be used directly as a leaf certificate. If the bit is not set, the certificate can only be used as a leaf certificate. If used as a parent with the bit not set, the certificate validation will fail. A certificate without an explicit ARExtension, i.e. inheriting the access rights from the KeyID.CARootKey can also act as a parent certificate.

Subsequent tables further detail the access right encodings for the different ARTypes.

7.6.4.1 PICC/Application level access rights

Table 20. Application access rights, specified via AID

Field	Length	Value	Description
ARType	1	0x01/0x81	Access rights for application with given AID
AID	3	0x000000 .. 0xFFFFFF	3-byte AID of the primary or single application
ACMap	2	see Table 18	2-byte map of granted access conditions of the primary or single application (LSB first)
AID2	[3]	0x000000 .. 0xFFFFFF	[Optional,only for multiple application selection] 3-byte AID of the secondary application
ACMap2	[2]	see Table 18	[Optional,only for multiple application selection] 2-byte map of granted access conditions of the secondary application (LSB first)

In the case of application access rights, specified via AID, as defined in Table 20, the ARValue consist of the 3-byte AID followed by a 2-byte ACMap. The latter defines the actual access rights granted for that application, as further defined in Table 18.

The ACMap must only grant a subset of the access rights associated with the targeted KeyID.CARootKey. For multiple application selection, this structure is repeated for the secondary application with ACMap2. In this case, a KeyID.CARootKey in both the primary and secondary application must be addressed during the authentication. Both these keys must be associated with the same public key value and (if applicable) issuer name. The ACMap will encode the access conditions granted by the primary application, and therefore must only grant a subset of the access rights associated with KeyID.CARootKey in the primary application. The ACMap2 encodes the access conditions granted by the secondary application. It must therefore only grant a subset of the access rights associated with KeyID.CARootKey in the secondary application.

See also Section 7.10.2.2 for the exact rules applied during certificate validation.

When application access rights are specified via DFName, the structure is similar, but replaces the 3-byte AIDs by the variable length DFName encoding. This encoding is defined in Table 21. Depending on whether the DFName is the PICC DFName or not, the PICC (that is, AID = 0x000000) ACMap encoding applies or not.

Access rights, specified via DFName, only allow for single application access rights, as multiple application selection is always via 3-byte AID.

Table 21. Application access rights, specified via DFName

Field	Length	Value	Description
ARType	1	0x02/0x82	Access rights for application with given DFName
DFNameLen	1	0x01.. 0x10	Length of the subsequent DFName of the application.
DFName	DFNameLen	FullRange	DFName of the application
ACMap	2	see Table 18	2-byte map of granted access conditions of the application

Provided under NDA only
COMPANY CONFIDENTIAL - personal copy for:
Sergio Salata
Quintagroup GmbH
Do not copy, share or reproduce
5f607f79d-c2ff-40f2-90f8-dec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

The ACMap if applicable must only grant a subset of the access rights associated with the targeted KeyID.CARootKey.

7.6.4.2 Delegated Application Management access rights

The last type of access rights that can be granted is for Delegated Application Management, as described in Section 7.11.2. These access rights are defined in Table 22. They grant access for either application creation with CreateDelegatedApplication, or deletion with DeleteApplication. The further payload in the ARValue consists mainly of the relevant parameters for CreateDelegatedApplication, or just the AID for DeleteApplication. Next to this, optionally a UID can be specified, restricting the rights to a single card (supporting charge-per-use scenarios).

This type of certificate can be issued by a CA related to a KeyID.CARootKey at the PICC level that has bit 2 set in its AC bitmap, as outlined in Section 7.6.4.1 and Section 7.6.3. Also a delegated CA (holding certificate with ARType 0x83) may issue the ARType 0x03 certificate. It is for example, possible to create a parent certificate for a specific application creation without UID restriction, and then have leaf certificates that further restrict for a specific UID.

Table 22. Delegated Application Management access rights

Field	Length	Value	Description
ARType	1	0x03/0x83	Access rights for delegated application management
UIDLen	1	0x00, 0x07, 0x0A	Length of the subsequent UID (if present)
UID	[UIDLen]	FullRange	[Optional, present if UIDLen != 0x00] 7-byte or 10-byte UID
DAMOperation	1		Delegated operation
		0x00	Creation
		0x01	Deletion
DAMPayloadLen	1		Length of the subsequent DAMPayload
		0x0A..0x21	[if DAMOperation = Creation]
		0x03	[if DAMOperation = Deletion]
DAMPayload	10..33	see Section 8.6.3	[if DAMOperation = Creation] Relevant CreateDelegatedApplication parameters, i.e. AID || DAMSlotNo || DAMSlotVersion || QuotaLimit || KeySett1 || KeySett2 [|| KeySett3] [|| AKSVersion || NoKeySets || MaxKeySize || RollKey] [|| ISOFileID] [|| ISODFName]
	3	see Section 8.6.3	[if DAMOperation = Deletion] Relevant DeleteApplication parameters, that is, AID.

These certificates also allow to:

- execute following commands: GetApplicationIDs, GetDFNames, GetDelegatedInfo, GetKeySettings.
- execute commands requiring access condition 0x2.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.7 Card Memory and Configuration Management

MF3E(H)x3 is characterized by several version parameters that are listed in Section 7.7.2. In particular MF3E(H)x3 is identified by its VCUID that is specified in Section 7.7.1. Finally, the card can be configured for several features specified in Section 7.7.3. The available user memory of the card can be monitored and managed as specified in Section 7.7.4.

7.7.1 Card UID

MF3E(H)x3 unique identifier (VCUID) is a 7 or 10 byte serial number defined as UID in ISO/IEC 14443-3. This VCUID is generated by the manufacturer of the card and programmed into a locked part of the NV-memory. The programmed VCUID is not changeable.

The first byte of the VCUID, *UID0*, as defined in ISO/IEC 7816-6 AMD 1 holds the manufacturer ID that is 0x04 for NXP Semiconductors.

The remaining bytes of VCUID, that is, *UID1*- *UID6* for double size or 7-byte UID are chosen by the manufacturer to be unique.

Note that despite the fact that a specific implementation of MF3E(H)x3 might not support all ISO/IEC 14443-3 UID lengths, see Section 7.3, all features in this document are defined generically with regards to UID length.

7.7.1.1 Random ID

MF3E(H)x3 can be configured with SetConfiguration to use a Random ID for the anticollision procedure as specified in ISO/IEC14443-3 [4]. The Random ID is defined on 4 bytes as a single-size UID defined in ISO/IEC 14443-3. UID0 is defined as 0x08 and UID1 - UID3 are random.

7.7.1.2 GetCardUID Command

Reading the VCUID of a card as defined in Section 7.7.1 is possible with the command GetCardUID, described in in Section 8.3.6 and Figure 35.

No parameters are passed with this command.

Even if the random UID feature is activated GetCardUID returns the real VCUID of the card. The default return format, see Section 8.3.6, holds a 7-byte UID. For other lengths, the returned (plain) byte string is formatted as follows:

0x00 || VCUID length || VCUID.

GetCardUID is allowed only if a previous successful authentication has been done with any key allowing authentications present on the card.

The communication mode is always CommMode.Full. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.7.1.3 Fast confidential UID retrieval with ISOSelectFile

MF3E(H)x3 supports fast confidential UID retrieval with ISOSelectFile. In the case of random ID configuration, this feature avoids an additional authentication with a nondiversified key to retrieve the UID with GetCardUID for applications using symmetric authentication.

The feature needs to be explicitly enabled for an application, using SetConfiguration 0x04 Bit 6, specifying a KeyID.AppUIDENCKey. Note that this cannot be combined with the ISOSelectFile integrated ECC-based authentication, as described in Section 7.5.6.6. Also if enabled, this feature gets precedence over a potential FCI configured in File 0x1F, see Section 7.17.3.4. The feature is not supported at PICC level.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Original Solutions GmbH
Do not copy, share or reproduce
5fc77790-c22c-405c-90fe-6ec3ca842f1

Once enabled, if the application is targeted for selection with ISOSelectFile, the FCI will be constructed as specified in Table 23, using BER-TLV encoding with short form lengths according to [11].

Table 23. ISOSelectFile response data

Tag	Length	Value
0x6F	0x22	File Control Information
0x85	0x20	Proprietary information not encoded in BER-TLV, consisting of: • RndIV (16 byte): random IV • Payload (16 byte): *E(KAppUIDENC, RndIV ⊕VCData) with VCData =(INFO* || *PDCap1* || *VCUID* || *ZeroPadding)*

The *Info* (1 byte) sent by the PD is the combination of the encoding of the length of the VCUID and a VC Type Identifier. PDCap1 (2 byte) defines the capabilities of the VC for the rest of the VC Selection protocol and the remaining transaction. This can be configured withSetConfiguration 0x05. If there is no application specific configuration, the PICC values will be taken. For more details, see Section 7.20. *Zeropadding* consists of the number of zero bytes needed to ensure VCData has a length of 16 bytes. Encryption is done with the KeyID.AppUIDENCKey according to CBC mode of NIST SP800-38A [2] using the *RndIV* as IV.

This FCI construct is equivalent to the first message of the Virtual Card Selection protocol supported by MIFARE DESFire from EV2 onwards. The full VC Selection protocol is not supported by MF3E(H)x3.

The integrity of the UID and returned capabilities is not ensured by the ISOSelectFile response itself, but for the UID, this can be acquired indirectly by, for example, using the UID in a subsequent key derivation and authentication.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.7.2 Card Version

MF3E(H)x3 is characterized by manufacturer-related data. These data are composed from HW-related information, SW-related information, and production-related information as detailed in Table 24.

Table 24. Manufacturer characteristics used as card version

Manufacturer Characteristics	Size in byte	Details
Hardware-related information:		
Vendor ID	1	Identification of the card vendor, 0x04 for NXP Semiconductors.
HW type	1	Hardware platform type.
HW subtype	1	Hardware platform subtype.
HW major version number	1	Hardware platform major version number.
HW minor version number	1	Hardware platform minor version number.
HW storage size	1	Hardware platform storage size. The 7 MS Bits (= n) code the storage size itself based on 2^n. The LS Bit is set to '0' if the size is exactly 2^n and to '1' if the storage size is between 2^n and $2^{(n+1)}$.
HW protocol	1	Hardware communication protocol type.
Software-related information:		
Vendor ID	1	Card vendor identification.
SW type	1	Card software type.
SW subtype	1	Card software subtype.
SW major version number	1	Card Software major version number.
SW minor version number	1	Card Software minor version number.
SW storage size	1	Card Software storage size. The 7 MS Bits (= n) code the storage size itself based on 2^n. The LS Bit is set to '0' if the size is exactly 2^n and set to '1' if the storage size is between 2^n and $2^{(n+1)}$.
SW protocol	1	Card Software communication protocol type.
Production-related information:		
VCUID	7	Card unique identifier as defined in Section 7.7.1. If the random UID is activated always 7 0x00 bytes are returned. Note that when switching ot random ID this is only reflected after reset and re-activation. The default return format, see Table 124, holds a 7-byte UID. For other lengths, the returned (plain) byte string is formatted as follows: 0x00 || VCUID length || VCUID.
Batch number	3	Production batch number.
Type ID	2	Type Identifier
CW of production	1	Bit 7: RFU
		Bit 6-0: Calendar week of card production in BCD coding (that is, week 36 is coded as 0x36)
Year of production	1	Bit 7: RFU
		Bit 6-0: Year of card production in BCD coding (i.e. year 2012 is coded as 0x12)

Provided under NDA only
COMPANY CONFIDENTIAL - personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07f2d-02c-4b-ac-90b-6ecc3caefcf1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.7.2.1 GetVersion Command

Reading the version of a card as defined in Section 7.7.2 is possible with the command GetVersion as defined in Section 8.3.5 and Figure 34.

The version data is return over 3 frames. The 1st Frame returns the hardware-related information, the 2nd returns the software-related information, and the 3rd and last frame returns the production-related information. (The version data is specified in Table 24.)

If there is an active authentication the command GetVersion requires CommMode.MAC. Information on the authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

One optional parameter, the Option byte, can be passed with this command. The Option byte indicates whether the Fab Identifier (representing the manufacturing site of the IC) shall be returned in the in GetVersionPart3 response. This optional parameter can only be set for MF3E(H)x3 products using the SW Minor Version 0x01.

7.7.3 Card Configuration

Card configuration can be applied on both PICC and application level. Depending on the selected level, different configuration options are available. Therefore, the PICC or the targeted application should have been selected before performing a SetConfiguration command.

7.7.3.1 SetConfiguration Command

SetConfiguration is updating PICC or application configuration settings, as defined in Section 8.3.3 and Figure 32. The command consists of an option byte and a data field with a size depending on the option. The option byte specifies the nature of the data field content depending whether PICC or an application is currently selected. Some options specified by their option byte value are only applicable at PICC level and some others are applicable both at PICC level and application level as defined in Table 25. If an option is only applicable at PICC level then it should be considered as RFU for application level.

In the below tab “No change” references are used with configurations that are persistent. This means that the associated configuration is left as it is already in the card and its value is not changed.

Table 25. SetConfiguration options list

Option byte	Level	Field	Length/ Bit Index	Description
0x00	**PICC and application**		**Total:1**	**PICC or Application Configurations**
	PICC	PICCConfig	1	PICC Configurations
			Bit 7	RFU
			Bit 6	Reserved
			Bit 5	UID0 configuration for Random ID refer to Random ID. Random ID is disabled at delivery time.
				0: [if Bit 1 = ’0’] No change.
				1: [if Bit 1 = ’0’] Not allowed.
				0: [if Bit 1 = ’1’] Enable ISO compliant random ID (UID0 = 0x08).
				1: [if Bit 1 = ’1’] Enable legacy random ID compatible with DESFire EV1 (UID0 = 0x80).
			Bit 4-2	Reserved

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 25. SetConfiguration options list...*continued*

			Bit 1	UseRIDconfiguration for Random ID refer to Random ID. Random ID is disabled at delivery time.
				0: No change
				1: Enable random ID. See Bit 5 for UID0 configuration. Once enabled, Random ID cannot be disabled anymore, but the UID0 configuration can still be adapted.
			Bit 0	Disable command Format at PICC level only. It can be still used if a delegated application is selected, see Section 7.7.4.2. Command Format at delivery time.
				0: No change
				1: Disable Format at PICC level only, once disabled cannot be enabled again.
	Delegated Application	AppConfig	1	Application configurations
			Bit 7-1	RFU
			Bit 0	Disable command Format at application level. This configuration can only be applied for delegated applications and is enabled at application creation.
				0: No change
				1: Disable Format for the selected delegated application.
0x01	**PICC**		**Total: 25, 33**	**DefaultKeys Update**
		PICCAppDefaultKey	25, 33	Update the KeyID.PICCAppDefaultKey. Data consists of 24 bytes (compatible with MIFARE DESFire) or 32 bytes of key and 1 byte of key version, formatted as Key_{data} || $Key_{version}$. For KeyType.AES128, only first 16 bytes of *Keydata* are used. If customized, conventional application creation targeting Key Type.AES256 is only allowed if a 32-byte *Keydata* has been set.
0x02	**PICC**		**Total: 1..20**	**ATS Update**
		UserATS	1..20	User defined ATS. The TL byte of the ATS should be as 0 < *TL* ≤ 20. FSCI 4 bits value part of T0 format byte (see ISO/IEC 14443-4) should be supported by the card. Default values of ATS are given in ATS. Refer to ISO/IEC 14443-4 for ATS definition.
0x03	**PICC**		**Total: 2**	**SAK Update**
		UserSAK	2	User defined SAK1 and SAK2 each of one byte long formatted as *SAK1* || *SAK2*. Default values of SAK1 and SAK2 are given in SAK. Refer to ISO/IEC14443-3 for SAK1 and SAK2 definition.
0x04	**PICC and Application**		**Total: 2, 3**	**Secure Messaging Configuration**
		SMConfigA	1	Secure messaging configuration (Byte A)
		SMConfigB	1	Secure messaging configuration (Byte B)
			Bit 7	RFU
			Bit 6	Enable ISOSelectFile fast UID retrieval, see Section 7.7.1.3. This option is only supported at application level.
				0: Disable (default)

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, distribute or reproduce
5fc07700-2ac-4bfc-90fe-ec3ca42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 25. SetConfiguration options list...*continued*

				1: Enable
			Bit 5	Enable free access over I^2C, see Section 7.6.1.
				0: Disable (default)
				1: Enable
			Bit 4	Enable EV Charging card-unilatteral authentication and related commands, see Section 8.14. This option is only supported at application level.
				0: Disable (default)
				1: Enable
			Bit 3	Enable ISOSelectFile integrated ECC-based authentication, see Section 7.5.7.
				0: Disable (default)
				1: Enable
			Bit 2	EV2 secure messaging configuration for FileType.StandardData.
				0: No change
				1: In VCState.AuthenticatedAES and VCState.Authenticated ECC, disable chained writing with WriteData in CommMode. MAC and CommMode.Full. If disabled at PICC level then it is disabled for every application of the card. None of these configurations are disabled by default. Once disabled they cannot be enabled again.
			Bit 1-0	Reserved
		CurveID	[1]	[Optional,present if SMConfig Bit 3 is set] Targeted curve
				0x0C: NIST P-256
				0x0D: brainpoolP256r1
		AppUIDENCKey	[1]	[Optional,present if SMConfig Bit 6 is set] UID encryption Key
				0x00 .. 0x0D: KeyID.AppUIDENCKey is the KeyID.AppKey to be used for UID encryption on fast confidential UID retrieval. ***Note:*** *The maximal key number depends on the number of application keys.*
0x05	**PICC and Application**		**Total: 10**	**CapabilityData**
				Capability data, consisting of PDCap1 and PDCap2. See Section 7.20 for details.
		RFU	1	RFU
		VCTID	1	VCTID Override: the VC Type Identified Override, as defined in Section 7.20. Note that as it is only 6 bits, the 2 most significant bits of this byte must be set to 0. This field contains the VCTID if the VC is of a type which PCDs may not recognize, while the VC is also backward compatible to an existing VCTID. The Operator can introduce new PDs in fields with VCs with a new VCTID even when this new VCTID is not supported by the PCDs. When this field is set, the PD communicates this value as VCTID rather than the built-in value. After the operator has upgraded the PCDs to support a higher level of VCTID, the operator can in the

Provided under NDA only
COMPANY CONFIDENTIAL, Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07796-022-4055-90fe-6ec8ca4211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 25. SetConfiguration options list...*continued*

				field set the VCTID Override back to 0x00 and the PD will use the built-in VCTID.
				0x00: No override done (default).
				0x04: MIFARE DESFire
				0x09: MF3E(H)x3
				other: currently not allowed. In future versions a valid value of backward compatible VCTID that the PD supports.
		RFU	1	RFU
		PDCap1.2	1	User configured PDCap1.2
		RFU	1	RFU
		PDCap2.2	1	Chip Manufacturer defined PDCap2.2: TransactionTimer, see Section 7.15.1.
				0x00: TransactionTimer disabled (default).
				0x01: TransactionTimer enabled with a threshold in the following range [1s , 3s].
				0x02: TransactionTimer enabled with a threshold in the following range [10s , 30s].
				0x03: TransactionTimer enabled with a threshold in the following range [100s , 300s].
				0x04..0xFF: RFU.
		RFU	2	RFU
		PDCap2.5	1	User configured PDCap2.5
		PDCap2.6	1	User configured PDCap2.6
0x06	**PICC and Application**		**Total: 17, 25**	**VCIID or application ISODFName renaming**
	PICC (17 byte)	VCIIDLen	1	VCIID length: must be in the range 0x01..0x10.
		VCIID	16	VCIID, padded with 0x00 bytes if needed. The default value is 0xD2760000850100 (a registered ISO DF name for MIFARE DESFire) which is recommended to be changed
	Delegated Application (25 byte)	DFNameLen	1	*ISODFName* length: must be in the range 0x01..0x10.
		DFName	16	*ISODFName*, padded with 0x00 bytes if needed
		DAMMAC	8	*DAMMAC*
	Conventional Application (17 byte)	DFNameLen	1	*ISODFName* length: must be in the range 0x01..0x10.
		DFName	16	*ISODFName*, padded with 0x00 bytes if needed
0x07				Reserved
0x08				Reserved
0x09				Reserved
0x0A				Reserved
0x0B				Reserved
0x0C	**PICC**		**Total: 2**	**ATQA Update**

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0770d-cac-A0c-90fc-6ecc3ce42f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 25. SetConfiguration options list...*continued*

		UserATQA	2	User defined ATQA, encoded LSB first as transmitted during ISO/IEC 14443 activation. Default values of ATQA are given in ATQA. Refer to ISO/IEC 144433 for ATQA definition.
0x0F	**PICC**		**Total: 3**	**NFC Management**
		NFCSupport	1	RFU
		ProtocolOptions	2	The crypto protocols supported over NFC. See Table 26. Default value is all protocols supported in the manufacturing features selection map are enabled.
0x10	**PICC**		**Total: 4**	**I2C Management**
		I2CSupport	1	RFU
		I2CAddress	1	The address used for the I^2C target (default 0x20)
		ProtocolOptions	2	The crypto protocols supported over I^2C. See Table 26. Default value is all protocols supported in the manufacturing features selection map are enabled.
0x11	**PICC**		**Total: 33**	**GPIO Management**
		GPIO1Mode	1	GPIO1 Mode
				0x00: disabled (default)
				0x01: input
				0x02: output
				0x03: Reserved
				0x04: down-stream power out
		GPIO1Config	1	GPIO1 Configuration, see Table 27.
		GPIO1PadCtrl	4	GPIO1 Pad Control, see Table 28.
		GPIO2Mode	1	GPIO2 Configuration
				0x00: disabled (default)
				0x01: input
				0x02: output
				0x05: output with NFCPause file
		GPIO2Config	1	GPIO2 Configuration, see Table 27.
		GPIO2PadCtrl	4	GPIO2 Pad Control, see Table 28.
		GPIO1Notif	1	GPIO notification on authentication. Note: notification shall only be allowed if GPIO1Mode is 0x02.
				0x00: disabled (default)
				0x01: enable authentication notification
				0x02: enable NFC field notification
		GPIO2Notif	1	GPIO notification on authentication. Note: notification shall only be allowed if GPIO2Mode is 0x02.
				0x00: disabled (default)
				0x01: enable authentication notification
				0x02:enable NFC field notification

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salad
Quantag IT Solutions GmbH
Do not copy, distribute or reproduce
5fc0770d-0ac405c-9016-0ecc3cae4251

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 25. SetConfiguration options list...*continued*

		ManageGPIO-AccessCondition	1	ManageGPIO access condition at PICC level
			Bit 7-6	RFU
			Bit 5-4	CommMode, see Table 7.
			Bit 3-0	Access Condition Value, see Table 17. Default 0xF.
		ReadGPIO-Access Condition	1	ReadGPIO access condition at PICC level
			Bit 7-6	RFU
			Bit 5-4	CommMode, see Table 7.
			Bit 3-0	AccessCondition Value, see Table 17.Default 0xF.
		AID	3	AID for additional access conditions and/or NFCPause file.
				0x000001.. 0xFFFFFF: any application different from the PICClevel.
		AddManageGPIO-AccessCondition	1	Additional ManageGPIO access condition within the application specified by AddAID.
			Bit 7-4	RFU
			Bit 3-0	AccessCondition Value, see Table 16. Default 0xF.
		AddReadGPIO-AccessCondition	1	Additional ManageGPIO access condition within the application specified by AddAID
			Bit 7-6	RFU
			Bit 5-4	CommMode, see Table 7.
			Bit 3-0	AccessCondition Value, see Table 16. Default 0xF.
		DefaultTarget	1	[Applicable when GPIO1Mode = 0x04] Targeted voltage/ currentlevel
				0x00: disable in rush current limit.
				0x01: power downstream of 1.8 V and 100 µA
				0x02: power downstream of 1.8 V and 300 µA
				0x03: power downstream of 1.8 V and 500 µA
				0x04: power downstream of 1.8 V and 1 mA
				0x05: power downstream of 1.8 V and 2 mA
				0x06: power downstream of 1.8 V and 3 mA
				0x07: power downstream of 1.8 V and 5mA
				0x08: power downstream of 1.8 V and 7 mA
				0x09: power downstream of 1.8 V and 10 mA
				0x0F: power downstream of 1.8 V and maximal available current
				0x11: power downstream of 2 V and 100 µA
				0x12: power downstream of 2 V and 300 µA
				0x13: power downstream of 2 V and 500 µA
				0x14: power downstream of 2 V and 1 mA
				0x15: power downstream of 2 V and 2 mA
				0x16: power downstream of 2 V and 3 mA

Provided under NDA only
COMPANY CONFIDENTIAL Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0770d-2ac-405c-90fe-ecc30ae4a7f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 25. SetConfiguration options list...*continued*

				0x17: power downstream of 2 V and 5 mA
				0x18: power downstream of 2 V and 7 mA
				0x19: power downstream of 2 V and 10 mA
				0x1F: power downstream of 2 V and maximal available current
		InRushTarget	1	[Applicable when GPIO1Mode = 0x04] Initial current limit to handle the in rush of current when charging external capacitor.
				0x00: disable in rush current limit.
				0x01: power downstream of 1.8 V and 100 µA
				0x02: power downstream of 1.8 V and 300 µA
				0x03: power downstream of 1.8 V and 500 µA
				0x04: power downstream of 1.8 V and 1 mA
				0x05: power downstream of 1.8 V and 2 mA
				0x06: power downstream of 1.8 V and 3 mA
				0x07: power downstream of 1.8 V and 5 mA
				0x08: power downstream of 1.8 V and 7 mA
				0x09: power downstream of 1.8 V and 10 mA
				0x11: power downstream of 2 V and 100 µA
				0x12: power downstream of 2 V and 300 µA
				0x13: power downstream of 2 V and 500 µA
				0x14: power downstream of 2 V and 1 mA
				0x15: power downstream of 2 V and 2 mA
				0x16: power downstream of 2 V and 3 mA
				0x17: power downstream of 2 V and 5mA
				0x18: power downstream of 2 V and 7 mA
				0x19: power downstream of 2 V and 10 mA
		InRushDuration	2	[Applicable when GPIO1Mode = 0x04] The duration to apply the InRushTarget
				0x0000.. 0xFFFF: targeted duration in ms.
		AdditionalCurrent	1	[Applicable when GPIO1Mode = 0x04] The additional current required by MF3E(H)x3 when supplying power downstream. Resolution: 0.4mA.
				0x00..0x1F
		NFCPauseFileNo	1	[Applicable when GPIO2Mode = 0x05] FileNo of the FileType.StandardData file for NFCPause
				0x00.. 0x1F: targeted FileNo
		NFCPauseOffset	3	[Applicable when GPIO2Mode = 0x05] Starting offset of the section within the targeted file triggering NFCPause
				0x000000.. 0xFFFFFF: targeted offset

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or upload
5fc07790-c02a-405c-9cfc-bec3bcae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 25. SetConfiguration options list...*continued*

		NFCPauseLength	3	[Applicable when GPIO2Mode = 0x05] Length of the section within the targeted file starting at NFCPauseOffset triggeringN FCPause
				0x000000.. 0xFFFFFF: targeted length
0x12				Reserved
0x13				Reserved
0x14				Reserved
0x15	**PICC and Application**		**Total: 2**	**Crypto API Management**
		Support	1	Crypto API Support
				0x02: ECDSA Sign Support
		CryptoAuthAC	1	Access condition for CryptoRequest
			Bit 7-6	RFU
			Bit 5-4	CommMode, see Table 7. Default CommMode.MAC.
			Bit 3-0	Access Condition Value, see Table 16 for application level and Table 17 for PICC level. Default 0x0.
0x16	**PICC and Application**		**Total: 6**	**Authentication Counter and Limit Configuration**
		Reserved	1	RFU
				0x00
		AuthCtrOption	1	Authentication counter options
			Bit 7-1	RFU
			Bit 0	AuthenticateEV2First (AES-based authentication)
				’0’: disabled (default)
				’1’: enabled
		AuthCtrLimit	4	Authentication Counter Limit
				0x00000000: AuthCtrLimit disabled
				0x00000001.. 0xFFFFFFFF: AuthCtrLimit enabled with the given value
0x17	**Application**		**Total: 4**	**HALT and Wake-up Configuration**
		WakeUpA	1	Wake-up options (Byte A)
			Bit 7	RFU
			Bit 6	GPIO wake-up: GPIO2 pull-down will trigger wake-up.
				’0’: disabled (default)
				’1’: enabled
			Bit 5-0	I^2C wake-up Address (default: 0x20): 6 msb of 7-bi I^2C address used for wake-up
				Full range
		WakeUpB	1	Wake-up options (Byte B)
			Bit 7	I^2C wake-up address (default: 0x20): lsb of 7-bit I^2C Wake-up Address

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07790-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 25. SetConfiguration options list...*continued*

				Full range
			Bit6-3	I^2C SDA wake-up cycles (default: 0x0): number of SDA cycles to wake-up
				Full range
			Bit 2	I^2C address wake-up: if targeted I^2C address matches the configured I^2C wake-up address, wake-up will be triggered.
				'0': disabled
				'1': enabled (default)
			Bit 1	I^2C SDA cycle wake-up: if I^2C SDA cycles match the configured I^2C wake-up cycles, wake-up will be triggered.
				'0': disabled (default)
				'1': enabled
			Bit 0	NFC field: wake up from HALT state on presence of NFC field.
				'0': disabled
				'1': enabled (default)
		RDACSetting	1	RDAC Setting: impacts how much energy is drawn from the RF field, while device is in HALT state.
				0x00..0xFF (default: 0x00)
		HALTOption	1	HALT options
			Bit 7-2	RFU
			Bit 1	GPIO2 reset: before entering power-saving HALT state, GPIO2 pin will reset to High-Z state.
				'0': disabled
				'1': enabled (default)
			Bit 0	GPIO1 reset : before entering power-saving HALT state, GPIO1 pin will reset to High-Z state.
				'0': disabled
				'1': enabled (default)
0x18 .. 0xFD				RFU
0xFE				Reserved
0xFF	**Application**		**Total: 3**	**Lock Configurations**
		LockMap	3	Bitmap where each bit encodes for the related configuration option if it is locked. LSB first, that is, first byte encodes Option 0x07-0x00.
			Bit 23-0	Lock bit
				'0': No Change
				'1': Lock configuration

Note: Application-specific remark to Table 25: *The InRush Target current shall always be set to the same level as the minimum configured power downstream current. The InRush current and it's ramp time determine*

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07704-92c-405c-90fe-6ecc3cae4211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

the system's settling time for buffer cap charging and initial ramp up of the IC external components. Therefore an InRushTarget configuration of 0x00 is causing system instabilities and shall be avoided.

The InRushTarget current and InRush Duration shall be chosen according minimum current needs of external components and the buffer cap size.

To configure the power downstream output current to maximum available current, that can be derived from the RF field, DefaultTarget shall be configured to 0x0F or 0x1F, for other configurations exceeding the maximum available current an error will be reported.

Changes of the output voltage level by changing DefaultTarget during already enabled power downstream does not have any impact, the initially configured output voltage is not changed.

The actual power downstream output current and output voltage provided at GPIO1 depends besides DefaultTarget configuration on the RF field strength and vary based on RF field changes and drop below 1.5V. These dynamic changes are not fully reflected when reading out the status via GPIO2.

Table 26. ProtocolOptions

Field	Length/ Bit Index	Description
ProtocolOptionsA	1	ProtocolOptions(Byte A)
	Bit 7	Reserved
	Bit 6-4	RFU
	Bit 3	ECDSASign (CryptoRequest_ECCSign, VDE_ECDSASign)
		’0’: Disabled
		’1’: Enabled (default)
	Bit 2	ECC-based Card-Unilateral Authentication (ISOInternalAuthenticate)
		’0’: Disabled
		’1’: Enabled (default)
	Bit 1	ECC-based Mutual and Reader-Unilateral Authentication (ISOGeneralAuthenticate and related commands)
		’0’: Disabled
		’1’:Enabled (default)
	Bit 0	AES-based Symmetric Authentication (AuthenticateEV2First)
		’0’: Disabled
		’1’: Enabled (default)
ProtocolOptionsB	1	ProtocolOptions(Byte B)
	Bit 7-0	RFU

Table 27. GPIOxConfig

Field	Length/ Bit Index	Description
GPIOxConfig	1	GPIOx Configuration
	-	**[if GPIOxMode is output (0x02 or 0x05)]**
	Bit 7-1	RFU

Provided under NDA only
CONFIDENTIAL - Personal copy for:
COMPANY CONFIDENTIAL
Sergii Salata
Quantum Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-495c-90fe-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 27. GPIOxConfig...*continued*

	Bit 0	Initial state after power off cycle
		0: Low (that is, equivalent to after CLEAR operation with ManageGPIO
		1: High (that is, equivalent to after SET operation with ManageGPIO
	-	**[else if GPIOxMode is down-stream power out (0x04)]**
	Bit 7-3	RFU
	Bit 2	AutoWTX: if enabled WTX will be sent on NFC Pause if power downstream is enabled.
		0: disabled
		1: enabled
	Bit 1	I2CSupport: if enabled, clock configuration is already adapted for future NFCPause I^2C communication on power downstream
		0: disabled
		1: enabled
	Bit 0	Backpower: if enabled backpower allows NFC communication (a.o.WTX) during power downstream.
		0: disabled
		1: enabled
	-	**[else]**
	Bit 7-0	RFU

Note: Application-specific remark to Table 27: *To enable stable NFC communication (NFCPause feature incl WTX handling) while power downstream is enabled, GPIOxConfig bit 0 (Backpower) must be set to '1' (enabled) and the reference design recommendation for capacitors on VCC and GPIO1 pins followed.*

Table 28. GPIOxPadCtrl

Field	Length/ Bit Index	Description
GPIOxPadCtrlA	1	GPIOx Pad Control (Byte A)
	Bit 7-2	RFU
	Bit 1-0	Debounce Filter value: the 2 MS bits of the 10-bit debounce filter value.
GPIOxPadCtrlB	1	GPIOx Pad Control (Byte B)
	Bit 7-0	Debounce Filter value (Resolution = 0.1 µs): the LS 8 bits of the 10 bit debounce filter value. Bit 0 is the LSB. Writing a value of 1 filters out glitches less than 0.1 µs. Writing a value of 1000 (over the 10 bits) filters out glitches less than 100 µs.
GPIOxPadCtrlC	1	GPIOx Pad Control (Byte C)
	Bit 7-3	RFU
	Bit 2	Debounce filter
		0: Disable debounce filter
		1: Enable debounce filter of min. 5 µs/max. 60 µs
	Bit 1-0	Input filter selection
		00: unfiltered input selected (filter of 50 ns selected but has no effect)

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantum Solutions GmbH
Do not copy, share or reproduce
5f6079d-02c-405c-90fe-6ecc3ca42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 28. GPIOxPadCtrl...*continued*

		01: unfiltered input selected (filter of 10 ns selected but has no effect)
		10: ZIF filtered input selected, filter of 50 ns selected
		11: ZIF filtered input selected, filter of 10 ns selected
GPIOxPadCtrlD	1	GPIOx Pad Control (Byte D)
	Bit 7-5	Input configuration
		000: Plain input with weak pull-up
		001: Plain input with repeater (bus keeper)
		010: Plain input
		011: Plain input with weak pull-down
		100: Weak pull-up
		101: Weak pull-down (*DISABLE_WPDN*)
		110: High impedance (analog I/O)
		111: Weak pull-down (*DISABLE_WPD*)
	Bit 4-1	Output configuration
		0000: I^2C S/F and FP transmit mode (SDA and SCL) and I^2C HStransmit mode (only SDAH)
		0001: I^2C HS transmit mode (only SCLK)
		0010: *I2C_TX_SFFP*
		0011: *I2C_TX_HS_SCLK*
		0100: GPIO Low speed mode (*GPIO_LOW_SPEED*_1)
		0101: GPIO Low speed mode (*GPIO_LOW_SPEED*_2)
		0110: GPIO High speed mode (*GPIO_HIGH_SPEED*_1)
		0111: GPIO High speed mode (*GPIO_HIGH_SPEED*_2)
		1000-1111: Output disabled
	Bit 0	Supply selection
		0: 1V8 signaling in I^2C mode
		1: 1V1 and 1V2 signaling in I^2C mode

The command SetConfiguration requires an active authentication.

The Table 29 shows for each of the supported options which key is required.

Table 29. SetConfiguration key requirements

Option byte	Key
0x00	KeyID.PICCMasterKey (PICC level) KeyID.AppMasterKey (Application level)
0x01	KeyID.PICCMasterKey (PICC level) KeyID.AppMasterKey (Application level)
0x02	KeyID.PICCMasterKey (PICC level)
0x03	KeyID.PICCMasterKey (PICC level)

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 29. SetConfiguration key requirements...*continued*

0x04	KeyID.PICCMasterKey (PICC level) KeyID.AppMasterKey (Application level)
0x05	KeyID.VCConfigurationKey (PICC level) KeyID.AppMasterKey (Application level)
0x06	KeyID.VCConfigurationKey (PICC level) KeyID.AppMasterKey (Application level)
0x0C	KeyID.PICCMasterKey
0x0F	KeyID.PICCMasterKey
0x10	KeyID.PICCMasterKey
0x11	KeyID.PICCMasterKey
0x15	KeyID.PICCMasterKey (PICC level) KeyID.AppMasterKey (Application level)
0x16	KeyID.PICCMasterKey (PICC level) KeyID.AppMasterKey (Application level)
0x17	KeyID.PICCMasterKey
0xFF	KeyID.PICCMasterKey (PICC level) KeyID.AppMasterKey (Application level)

The command SetConfiguration requires CommMode.Full. Information on authentication and secure messaging dependent structure of the command can be found in Section 8.2.

The command is rejected if:

- required authentication is not active.
- a PICC only option is given and the PICC level (AID 0x000000) is not selected.
- option 0x00
  - application level variant when a conventional application is selected.
- updating with option 0x02 the ATS
  - with a TL equal to 0 or strictly bigger than 20.
  - with T0 containing an FSCI > 0x8, as the PICC supports up to 256 byte command size, see Section 7.3.1.3.
- option 0x04
  - (at PICC level) enabling EV Charging support with bit 4.
  - (at PICC level) enabling free access over I^2C with bit 5.
  - (at PICC level) enabling fast confidential UID retrieval with bit 6.
  - (at application level) enabling both ISOSelectFile integrated ECC-based authentication and fast confidential UID retrieval with bit 3 and bit 6.
- option 0x05
  - updating capability data at application level while this has not been enabled at application creation
  - writing PDCap22 to a value different than 0x00 at PICC level where the TransactionTimer is not supported.
  - writing invalid VCTID.
- updating with option 0x06 the VCIID or *ISODFName* while
  - requiring unsupported length
  - providing inconsistent padding
  - targeting application that was created without *ISODFName*

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

– targeting delegated application and providing invalid DAMMAC
- option 0x15
  – issued at application level while application specific configuration was not enabled at application creation.

When configuring the ID with Option 0x00 Bit 0, Bit 5 and/or Bit 6, a change will only take place after the next power off reset.

When configuring the SAK with Option 0x03, MF3E(H)x3 shall accept any value, but:

- bit 2 of SAK1 (that is, b3 according to the numbering in [5]) is ignored and always set to '1' indicating UID incomplete.
- bit 2 of SAK2 (that is, b3 according to the numbering in [5]) is ignored and always set to '0' indicating UID complete.

A change in this configuration is guaranteed after the next power off reset.

When issuing option 0x6 at application level, if targeting a delegated application a *DAMMAC* must be presented (as defined in the table above). The *DAMMAC* is computed as described in Section 7.11.2.2.2. The *DAMMAC* is calculated with the following input:

$$DAMMAC = MACt_{DAM}(K_{PICCDAMMAC}, OldISODFNameLength \,||\, OldISODFName \,||\, NewISODFNameLength \,||\, NewISODFName)$$

With *OldISODFNameLength* being *NewISODFNameLength* being 8-bit length values. *OldISODFName* and *NewISODFName* are 16-byte strings. If the length is less than 16 bytes, the string is padded with 0x00 bytes. So the input to the MAC calculation is always 34 bytes. *OldISODFName* is the current *ISODFName* of the targeted application.

When configuring the ATQA with Option 0x0C, MF3E(H)x3 shall accept any value, but bit 7-6 of LSB (that is, b8-b7 according to the numbering in [5]) are ignored and set according to the actual UID length.

A change in this configuration is guaranteed after the next power off reset.

Option 0x0F and 0x10 configures the communication interfaces of the product, and what cryptographic protocols are available over each interface. Note that extreme care needs to be taken when configuring these options as disabling authentications on both interfaces will make the product unusable. Also, option 0x10 does not check the provided I2CAddress against reserved addresses as specified in [31].

Option 0x11 allows for configuring the GPIO pins and related access conditions for further managing and/or reading them, see also Section 8.11. Note that values related to specific modes are not checked for consistency. This means it is the user responsibility to provide a meaningful configuration. Even if not applicable for the configured mode, the provided values will be stored and returned by GetConfiguration. Next to the access conditions at PICC level, MF3E(H)x3 supports delegating the ManageGPIO and ReadGPIO access rights to one other application. If this is not required, addAID can be set to any value, while keeping both access conditions set to 0xF. Note that when configuring an NFCPauseFile, MF3E(H)x3 does not check if the file exists and is of a FileType.StandardData.

Option 0x15 configures some aspects of the generic Crypto API, see Section 7.15 . Access Condition defines the communication mode and required access rights for CryptoRequest. The default is CommMode.MAC and KeyID.PICCMasterKey or KeyID.AppMasterKey access rights. Note that this access right can be defined separately for each application if this was enabled at application creation through KeySett3 (see Section 8.6.1) Bit 2.

Option 0x17 configures aspects of HALT and related Wake-up Configuration. Note that HALT refers here to the internal power-saving state when VCC supplied, that is, not to be confused with ISO/IEC 14443 HALT state. Note that if GPIOx reset is disabled before entering HALT state, the GPIO pin will remain its state. However, when leaving the HALT state, the device goes through a PoR cycle, and thus the GPIO pins will at that moment go through the High-Z state.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Saez, Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fc0772d-32c8-9bf6-66cc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

The HALT current depends on the GPIO pin configuration. Floating GPIO inputs result in increased HALT currents. To avoid increased HALT currents due to floating GPIO inputs the following shall be considered:

- If GPIOwake-up is enabled internal pull-ups are activated and no floating in put behavior can occur.
- If GPIOwake-up is disabled and if bit 1 (reset) is disabled only with an external pull-up resistor increased HALT currents can be guaranteed.

NFC field and I^2C address wake-up are enabled by default. Note that NFC field wake-up being enabled means that as long as an NFC field is present, the MF3E(H)x3 will immediately wake-up again when entering HALT state. The RDAC configuration is only relevant if NFC field wake-up is disabled. If enabled, RDAC must be configured to 0x00. However, note that this is not checked by the implementation. The SDA cycle and GPIO wake-up are disabled by default.

Option 0xFF allows locking the other configurations. A bitmap *Lockmap*is to be provided. Note that this is sent LSB first, that is, to lock Option 0x00, Bit 0 of the first transmitted byte must be set. Setting a bit for a non-supported option (RFU or Reserved) does not have any effect. Once a configuration is locked, it cannot be unlocked, that is, setting a bit to 0 does not change the current state. This option is supported at both PICC-level and application-level.

7.7.3.2 GetConfiguration Command

GetConfiguration is retrieving card or application configuration settings, as defined in Section 8.3.4 and Figure 33.

If no configuration option byte is specified, then manufacturer data like the NXP Product Features Map is returned. The GetConfiguration is rejected at the PICC or application level depending on the option, see SetConfiguration. If no Option is provided, the command can be issued either at PICC or at application level. Option 0x01 is not supported as the KeyID.PICCAppDefaultKey cannot be outputted.

The GetConfiguration is subject to the same access restrictions as the SetConfiguration that is, it is subject to CommMode.Full, requiring KeyID.AppMasterKey access rights at application level, or KeyID.PICCMasterKey or KeyID.VCConfigurationKey access rights at PICC level.

7.7.4 Memory Management

The nonvolatile memory available for user data is allocated in blocks of 32 bytes. The complete user memory is available for application and file creation (including overhead). All PICC level keys and configuration data shall be located outside the user memory.

The supported memory configurations are defined in Table 30.

Table 30. Supported memory configuration

Memory Configuration	Size in bytes	Size in blocks
2 kB	2560	0x0050
4 kB	5120	0x00A0
8 kB	8192	0x0100
16 kB	16384	0x0200

Commands that impact the memory structure itself (for example creation or deletion of applications) activate an automatic mechanism that protects the application and file structure from getting corrupted. If the card is torn during command execution, it is ensured that on the next activation the memory structure is automatically updated so that the card behaves either exactly as it did before the command execution, or as it would have been after having completed a successful execution.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ccc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

The only exception is CreateDelegatedApplication, see also Section 7.11.2.2. If an existing delegated application is being replaced, the resulting card state after tearing may be such that the replaced application is deleted (similar as if it was deleted with DeleteApplication) and the new delegated application has not been created yet. Reissuing CreateDelegatedApplication is still possible to reach the intended state with the new delegated application being created.

7.7.4.1 Free memory with FreeMem command

The available free user memory on the card is returned with FreeMem as defined in Section 8.3.1 and Figure 30.

No parameters are passed with this command.

The memory size in bytes available is returned as an unsigned integer.

If the PICC is authenticated, then the command FreeMem requires CommMode.MAC. Otherwise, FreeMem is transmitted in plain. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.7.4.2 Format memory with Format command

User memory can be formatted and released with Format as defined in Section 8.3.2 and Figure 31.

No parameters are passed with this command.

The behavior of the command depends on the selected application.

If no application is selected, that is, the PICC level is selected (AID 0x000000), Format releases all allocated user memory on the PICC. All applications are deleted and all files within those applications are deleted. If delegated applications have been created on the card, all memory reservations for *QuotaLimit* and the *DAMSlot* associations are also erased. The memory is released and can be reused. The PICC level keys, and related key settings are not influenced by this command, so they keep their currently set values. This variant of the command can be disabled using the command SetConfiguration. This command always requires a preceding authentication with the KeyID.PICCMasterKey.

If a single application is selected, the command is rejected if this application is not a delegated application, see Section 7.11.2. If a delegated application is targeted, Format releases all allocated file memory in the application. All files are deleted. The memory is released and can be reused. The application keys and the application configuration settings keep their currently set values. This command does not influence them. However, as all the files are deleted, so is the FileType.TransactionMAC file (if present) and the related KeyID.AppTransactionMACKey command, which is also erased. This command always requires a preceding authentication with the KeyID.AppMasterKey.

If multiple applications are selected, see Section 7.11.3, the primary application is targeted. Therefore, this command always requires a preceding authentication with the KeyID.AppMasterKey of the primary application. This variant of the command can be disabled for individual applications using SetConfiguration.

The command Format requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

Provided under NXP COMPANY CONFIDENTIAL - personal copy for: Sergii Ivanov, Quantum Systems GmbH Do not copy, share or reproduce 5fc0779d-c22c-4bbc-901e-ec3cee421f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.8 Symmetric Key management

This chapter covers the key management aspects of symmetric keys for MF3E(H)x3. The key management of asymmetric keys is detailed in Section 8. At any place in this chapter where ”key‘(s)“ is mentioned without and explicit qualification, symmetric key(s) are meant.

MF3E(H)x3 supports a set of PICC level keys, see Section 7.10.1.2, related with PICC level functionality like PICC level configuration, (delegated) application creation and proximity check.

Each application can also contain one or multiple sets of application keys, see Section 7.8.5, related with application level functionality like application key management, file creation and deletion, and file access.

7.8.1 Key Hierarchy

MF3E(H)x3 associates symmetric keys to and application levels as shown in Figure 16. Those keys are persistent and stored in NVM. Next to its value, each key is characterized by two additional properties:

- Key type
- Key version

These two properties are described respectively in Section 7.8.2 and Section 7.8.3. The type property determines the key size and how the key is stored in NVM. While the key version helps to keep track of key values present on the PICC.

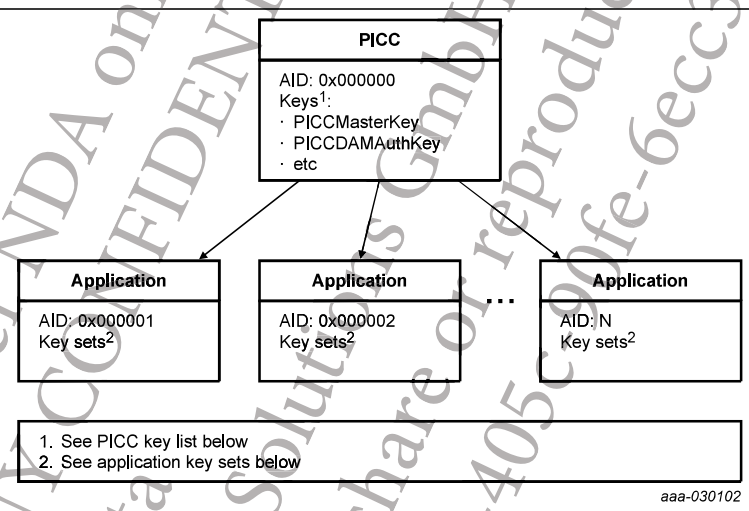


Figure 11. Key architecture

7.8.1.1 Key Protection

Any key managed by MF3E(H)x3 cannot be retrieved with the command interface.

Keys are protected internally by MF3E(H)x3 key management functionality. The MF3E(H)x3 implementation will ensure key confidentiality and integrity. The specification of these mechanisms is not linked to the interface and therefore out of the scope of this document.

7.8.1.2 Session Keys

Session keys are temporary keys created in the card, but are not persistent and therefore are not described in this chapter. Throughout the document, they get the prefix ’Ses’ for their naming: KeyID.SesXXX or K_{SesXXX}.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07790-c22c-405c-90fe-6ecc3cae42f1

7.8.2 Key Types

MF3E(H)x3 supports three key types as defined in Table 31. As shown in the table, the different key types are represented by two bits.

Table 31. Supported key types

Key Type	Bit Representation	Description
Reserved	00b	Reserved
Reserved	01b	Reserved
KeyType.AES128	10b	AES-128 keys
KeyType.AES256	11b	AES-256 keys

This representation is used at several places in the document.

These two key types are considered to have different strength [24]. Increasing strength is according to the order in Table 31. Concretely, this means KeyType.AES256 is considered stronger than KeyType.AES128.

This is important for the key set rolling as defined in Section 7.8.5.1.4.

KeyType.AES and KeyType.AES256 keys are stored in resp. of 16 or 32 bytes and are handled according to [1].

7.8.3 Key Versioning

MF3E(H)x3 supports the versioning of the keys by relating with each key a 1 byte key version number. The version of any addressable key can be read using GetKeyVersion.

7.8.4 PICC Level

7.8.4.1 PICC Keys

PICC keys are defined in Table 32 and described after. These keys are used to maintain the PICC itself. Operations such as application management, memory management or key management may require a certain PICC key authentication.

The Table 32 defines the key identifier used throughout the document together with the related symbol used in formulas to refer to the related key value. The following columns define the number to be used to address the key over the command interface, the allowed key types, the key protecting the access to change this key and, finally, whether this key can be used to authenticate with.

Table 32. Keys at PICC level

Key Identifier	Symbol	Key Number	Key Type	Change Key	Auth
Addressable keys:					
KeyID.PICCMasterKey	$K_{PICCMaster}$	00h	KeyType.AES128 KeyType.AES256	KeyID.PICCMasterKey	yes
Reserved		01h … 04h			
KeyID.PICCDAMAuthKey	$K_{PICCDAMAuth}$	10h	KeyType.AES128 KeyType.AES256	KeyID.PICCMasterKey	yes
KeyID.PICCDAMMACKey	$K_{PICCDAMMAC}$	11h	KeyType.AES128 KeyType.AES256	KeyID.PICCMasterKey	no
KeyID.PICCDAMENCKey	$K_{PICCDAMENC}$	12h	KeyType.AES128 KeyType.AES256	KeyID.PICCMasterKey	no

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-622c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 32. Keys at PICC level...*continued*

Key Identifier	Symbol	Key Number	Key Type	Change Key	Auth
KeyID.NXPDAMAuthKey	$K_{NXPDAMAuth}$	18h	KeyType.AES128 KeyType.AES256	N/A	yes
KeyID.NXPDAMMACKey	$K_{NXPDAMMAC}$	19h	KeyType.AES128 KeyType.AES256	N/A	no
KeyID.NXPDAMENCKey	$K_{NXPDAMENC}$	1Ah	KeyType.AES128 KeyType.AES256	N/A	no
KeyID.VCConfigurationKey	$K_{VCConfiguration}$	20h	KeyType.AES128 KeyType.AES256	KeyID.PICCMasterKey or KeyID.VCConfigurationKey	yes
KeyID.VCProximityKey	$K_{VCProximity}$	21h	KeyType.AES128 KeyType.AES256	KeyID.VCConfigurationKey	no
Reserved		22h … 23h			
KeyID.ICUpgradeKey	$K_{ICUpgrade}$	30h	KeyType.AES128 KeyType.AES256	N/A	no
Non-addressable key:					
KeyID.PICCAppDefaultKey	$K_{PICCAppDefault}$	N/A	N/A	KeyID.PICCMasterKey via SetConfiguration	no

The key number defined in Table 32 is valid for addressing at PICC level, that is, when AID 0x000000 is selected. Another numbering scheme is used at application level to address the application keys.

Non-supported or reserved key number values at PICC level, if targeted in a command, will be answered with NO_SUCH_KEY.

At delivery the following PICC level keys are disabled: KeyID.PICCDAMKeys, KeyID.NXPDAMKeys and the KeyID.VCConfigurationKey and KeyID.VCProximityKey. Disabled PICC level keys cannot be used for any cryptographic purpose. A command trying to access any key while the key is still disabled, will be rejected. The behavior of the KeyID.VCProximityKey is defined in Section 7.18.

Note that KeyID.PICCAppDefaultKey is not addressable in this way. It is managed differently via SetConfiguration as it is used to initialize application keys of all key types.

7.8.4.1.1 PICCMasterKey

The KeyID.PICCMasterKey ($K_{PICCMaster}$) is the master key of the PICC. It is used to restrict access to the following functionality at PICC level (that is, AID 0x000000 selected):

- Key updating for some of the keys, see Table 32.
- PICC configuration with SetConfiguration and ChangeKeySettings.
- Formatting with Format.
- Depending on PICCKeySettings, application creation and deletion and application information retrieval, see Section 7.8.4.2.

At delivery, the command KeyID.PICCMasterKey is set to the default value of all zero bytes for key value and version.

Key Type: The KeyID.PICCMasterKey can be a KeyType.AES128 or KeyType.AES256 key. The key type is set when changing KeyID.PICCMasterKey with ChangeKey or ChangeKeyEV2. This key type also defines the type of the Delegate Application Management keys, see Section 7.8.4.1.2. At delivery, KeyID.PICCMasterKey will have the KeyType.AES128 key type.

Provided under DA only
CONFIDENTIAL - Personal copy for:
Georgii Salatov
Quantas IT Solutions GmbH
Do not copy, share or reproduce
5fc07791-22c-405c-9065-6ee3ae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Key Enable: The command KeyID.PICCMasterKey is always enabled.

Key Change: The KeyID.PICCMasterKey is changeable with ChangeKey or ChangeKeyEV2 with an active authentication with KeyID.PICCMasterKey. KeyID.PICCMasterKey can be frozen depending on the PICC key update setting defined in Table 33.

Authentication: The command KeyID.PICCMasterKey is available for authentication.

7.8.4.1.2 PICCDAMKeys

The KeyID.PICCDAMAuthKey ($K_{PICCDAMAuth}$), KeyID.PICCDAMENCKey ($K_{PICCDAMENC}$) and KeyID.PICCDAMMACKey ($K_{PICCDAMMAC}$) are keys associated with Delegated Application Management specified in Section 7.11.2. They are named as well KeyID.PICCDAMKeys.

Key Type: The KeyID.PICCDAMKeys share key type as KeyID.PICCMasterKey. If the key type of KeyID.PICCMasterKey changes then the KeyID.PICCDAMKeys change also to the same new type as KeyID.PICCMasterKey.

Key Enable: The KeyID.PICCDAMKeys are disabled by default. On a key per key basis they are enabled only when they are written with a ChangeKey or ChangeKeyEV2. If the key type of KeyID.PICCMasterKey changes then KeyID.PICCDAMAuthKey, KeyID.PICCDAMMACKey and KeyID.PICCDAMENCKey are disabled. They must be written again to be enabled. If KeyID.NXPDAMKeys are enabled with PICCKeySettings Bit 4, KeyID.PICCDAMKeys are disabled. Independently whether the KeyID.PICCDAMKeys had already been changed, they are required to be changed again for enabling them if Bit 4 is set back to 0.

Key Change: The KeyID.PICCDAMKeys are changeable with ChangeKey or ChangeKeyEV2 with an active authentication with KeyID.PICCMasterKey. The KeyID.PICCDAMKeys cannot be changed while the KeyID.NXPDAMKeys are enabled.

Authentication: The KeyID.PICCDAMAuthKey is available for authentication. KeyID.PICCDAMMACKey and KeyID.PICCDAMENCKey are not available for authentication.

7.8.4.1.3 NXPDAMKeys

The KeyID.NXPDAMAuthKeys ($K_{NXPDAMAuth}$), KeyID.NXPDAMENCKey ($K_{NXPDAMENC}$) and KeyID.NXPDAMMACKey ($K_{NXPDAMMAC}$) are keys associated with Delegated Application Management specified in Section 7.11.2. They allow Delegated Application Management by NXP, for example, through the MIFARE 2GO service. They are named as well KeyID.NXPDAMKeys. Note that in the generic text, if it is not relevant to distinguish KeyID.NXPDAMKeys and KeyID.PICCDAMKeys, the KeyID.PICCDAMKeys are referenced.

The KeyID.NXPDAMKeys are written into the chip during manufacturing. All three KeyID.NXPDAMKeys are diversified, which means they are derived from a master key using the UID. This is done as described in Section 7.22.3.

Key Type: The KeyID.NXPDAMKeys are all KeyType.AES128 or KeyType.AES256. The key type is decided during trust-provisioning and cannot be changed afterwards.

Key Enable: The KeyID.NXPDAMKeys are disabled by default. They can be enabled by setting PICCKeySettings Bit 4.

Key Change: The KeyID.NXPDAMKeys are not changeable.

Authentication: The KeyID.NXPDAMAuthKeys is available for authentication. However, note that only AuthenticateEV2First and AuthenticateEV2NonFirst are supported. KeyID.NXPDAMMACKey and KeyID.NXPDAMENCKey are not available for authentication.

Provided under NDA
COPYRIGHTED MATERIAL
Seoul Semiconductor Solutions GmbH
Quantum Solutions
Do not copy, share or reproduce
5fc07799-c92c-4956-90fe-fee3ceb9ff

7.8.4.1.4 VCConfigurationKey

The KeyID.VCConfigurationKey ($K_{VCConfiguration}$), supports changing the KeyID.VCProximityKey and configuring SetConfiguration Option 0x05 and 0x06.

Key Type: The KeyID.VCConfigurationKey is an KeyType.AES128 or KeyType.AES256 key. The key type is set when changing

KeyID.VCConfigurationKey with ChangeKey or ChangeKeyEV2.

Key Enable: The KeyID.VCConfigurationKey is disabled by default. It is enabled when written with a ChangeKey or ChangeKeyEV2. Once enabled, it cannot be disabled again.

Key Change: The KeyID.VCConfigurationKey is changeable with ChangeKey or CChangeKeyEV2 with an active authentication with KeyID.PICCMasterKey or KeyID.VCConfigurationKey itself. Note that to enable KeyID.VCConfigurationKey, an authentication with KeyID.PICCMasterKey will be required to change this key for the first time.

Authentication: The KeyID.VCConfigurationKey is available for authentication.

7.8.4.1.5 VCProximityKey

The KeyID.VCProximityKey ($K_{VCProximity}$) is used for Proximity Checking specified in Section 7.18. An application can have its own KeyID.VCProximityKey. This must be enabled at application creation with CreateApplication or CreateDelegatedApplication. The KeyID.VCProximityKey is KeyType.AES128 or KeyType.AES256 key. The key type is set when changing KeyID.VCProximityKey with ChangeKey or ChangeKeyEV2.

Key Type: The KeyID.VCProximityKey is an KeyType.AES128 or KeyType.AES256 key. The key type is set when changing

KeyID.VCProximityKey with ChangeKey or ChangeKeyEV2.

Key Enable: The KeyID.VCProximityKey is disabled by default. It is enabled when written with a ChangeKey or ChangeKeyEV2. Once enabled, it cannot be disabled again.

Key Change: The KeyID.VCProximityKey is changeable with ChangeKey or ChangeKeyEV2 with an active authentication with KeyID.VCConfigurationKey.

Authentication: The KeyID.VCProximityKey is available for authentication.

7.8.4.1.6 PICCAppDefaultKey

The KeyID.PICCAppDefaultKey ($K_{PICCAppDefault}$) holds the key initialization value and version used in an application creation. The keys of the created application are set with the key data of the KeyID.PICCAppDefaultKey. This holds for all conventional applications created with CreateApplication.

Applications created through delegated application management (see Section 7.11.2) with CreateDelegatedApplication use the KeyID.AppDAMDefaultKey to set the initialization value of their keys.

Key Type: The KeyID.PICCAppDefaultKey has no type associated to it. Instead it consists of a 24-byte or 32-byte string for its value and a 1-byte key version. The 24-byte configuration, see SetConfiguration, is supported for compatibility with MIFARE DESFire.

On key initialization, the key value used depends on the key type of the targeted application. For KeyType.AES128, only the left most 16 bytes are used. For KeyType.AES256, this is only possible if a 32-byte value has been set and the full 32 bytes are used. If the KeyID.PICCAppDefaultKey has been customized with 24-byte string, it is not possible to create KeyType.AES256 conventional applications with CreateApplication. By default, at delivery the KeyID.PICCAppDefaultKey is initialized with 32-byte all zero key value and zero key version.

Provided under NDA on a CONFIDENTIAL - Personal copy for: Copyright Sales Solutions GmbH Do not copy, share or reproduce 5fc5779d-c224-405c-90fe-6ecc2ce42f1

Key Enable: The KeyID.PICCAppDefaultKey is always enabled.

Key Change: The KeyID.PICCAppDefaultKey is changeable with SetConfiguration as specified in Section 7.7.3.1.

Authentication: The KeyID.PICCAppDefaultKey is not available for authentication.

7.8.4.2 PICCKeySettings

The PICC holds as well a key setting defining the PICC key use. The PICCKeySettings are defined on 1 byte and are specified in Table 33. The commands GetKeySettings and ChangeKeySettings are used to retrieve and update the PICCKeySettings.

Table 33. PICCKeySettings byte definition

PICCKeySettings	Bit Index	Description	Value
RFU	Bit 7		
RFU	Bit 6		
RFU	Bit 5		
Enable KeyID.NXPDAMKeys	Bit 4	KeyID.NXPDAMKeys are	0(default): disabled. KeyID.PICCDAMKeys can be enabled and used.
			1: enabled. KeyID.PICCDAMKeys cannot be used, only changed.
PICCKeySettings changeable	Bit 3	PICCKeySettings is	0: frozen
			1 (default): changeable with KeyID.PICCMasterKey authentication.
Application create/delete configuration	Bit 2	KeyID.PICCMasterKey authentication is	0: required for access to CreateApplication and DeleteApplication.
			1 (default): not required for access to Create Application. DeleteApplication requires KeyID.PICCMasterKey or KeyID.App MasterKey authentication.
Directory access configuration	Bit 1	KeyID.PICCMasterKey or KeyID.PICCDAMAuthKey authentication is	0: required for access to GetApplicationIDs, Get DFNames, GetKeySettings and GetDelegatedInfo commands.
			1 (default): not required for access to Get ApplicationIDs, GetDFNames, GetKeySettings and GetDelegatedInfo commands.
KeyID.PICCMasterKey changeable	Bit 0	KeyID.PICCMasterKey is	0: frozen
			1 (default): changeable with KeyID.PICCMasterKey authentication.

7.8.5 Application Level

Application keys are associated with one application. Each application holds its own keys. They are addressed as soon as the application is selected.

Regular KeyID.AppKeys are grouped in key sets, see Section 7.8.5.1.

Application keys are used for the file access control. Some of them can have additional roles assigned, like being required for key changing. Next to this, an application can also have its own KeyID.VCProximityKey for

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag TI Solutions GmbH
Do not copy, share or reproduce
5fd179d-c22c-40be-aafe-6ec3ca42f1

proximity check; and contains some none-addressable keys. An overview is given in Section 7.8.5.2. These roles and other key related configurations are defined via the key settings, see Section 7.8.5.3.

7.8.5.1 Key Sets

Application keys are grouped into key sets. An application can hold up to 16 key sets. The number of key sets is fixed at application creation and it is not possible to change it thereafter. If the key set parameters are not present at application creation, then the application is created with a single key set. It is also possible to have no keys and as a consequence no key sets for an application.

A key set holds a maximum of 14 keys. The number of keys is fixed at application creation. It is the same for all key sets. The keys are absolutely numbered from 0 to 13 within the key set as shown in Figure 12.

If the application has no keys, the application key setting specified in Section 7.8.5.3 should be set adequately for the application to be usable.

Application key architecture with several key sets is shown in Figure 12.

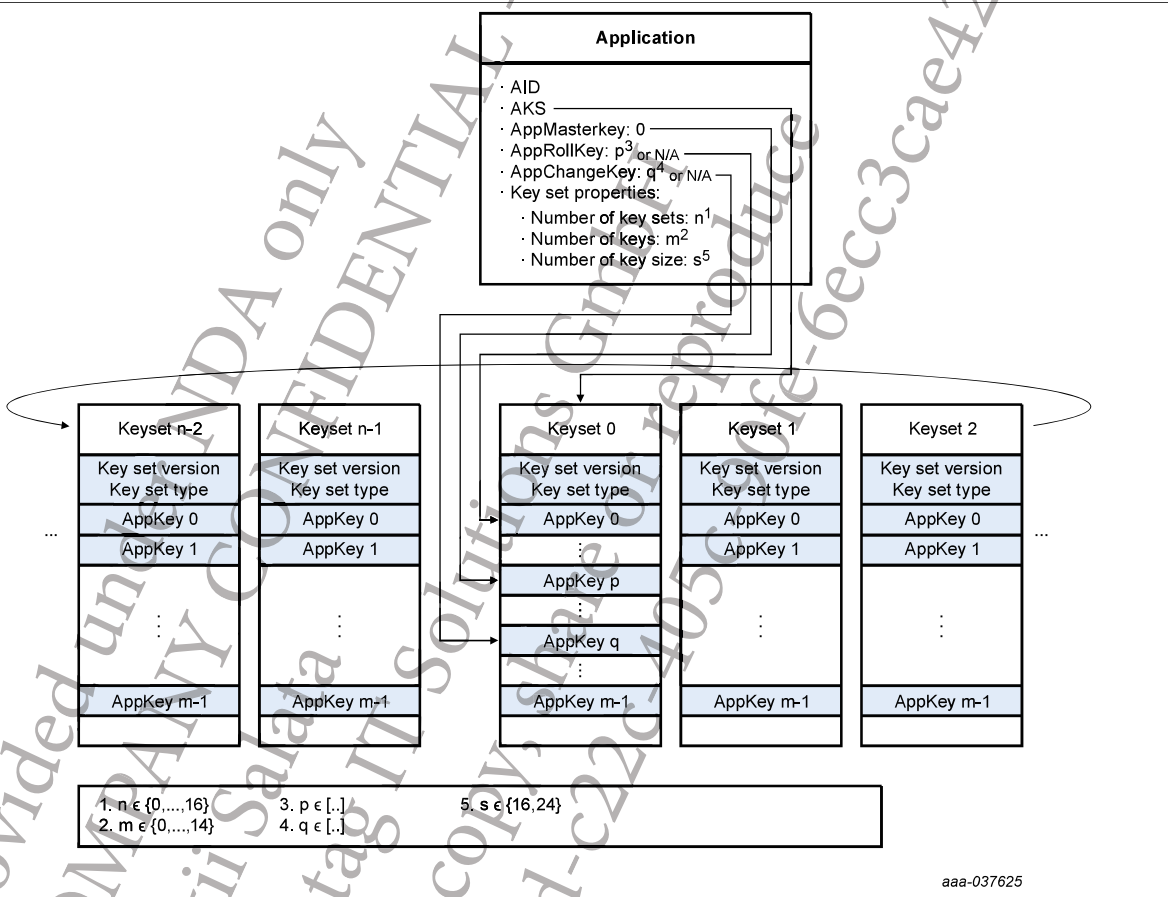


Figure 12. MF3E(H)x3 application key architecture

7.8.5.1.1 Active Key Set

One key set within an application is called the active key set or AKS. Only keys from the active key set can be used for authentication. The active key set can be changed to point to another key set of the same application by executing a so called key set rolling described in Section 7.8.5.1.4.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

The AKS is addressed by key set number 0x00, so key set addressing is done relatively to the AKS.

One can see in Figure 17 that the AKS is pointing on key set with address number 0. Then the number of each key set belonging to this application is increment by 1 to the right cycling back to the left when the last key set is reached. When the AKS is updated to point to another key set, the address numbering is changed and updated relatively to the new AKS.

7.8.5.1.2 Key Set Properties

A key set is characterized by some properties defined at application level and some at key set level as defined in Table 34.

Key set properties linked at application level are fixed at application creation through CreateApplication or CreateDelegatedApplication. They are shared between all key sets of an application.

Key set properties linked at key set level are particular to each key set. They are set during key set personalization, as defined in Section 7.8.5.1.3 or for the AKS at application creation through CreateApplication or CreateDelegatedApplication>.

Table 34. Application key set properties

Key set properties	Linked with	Description
Number of key sets	Application	Number of key sets within this application.
Number of keys	Application	Number of keys within each key set.
Max key size	Application	Maximum key size possible for all key set. Either 16 bytes, limiting the key type to KeyType.AES128 or 32 bytes, allowing both KeyType.AES128 and KeyType.AES256 for this application.
Key set version	Key set	When initializing a key set with FinalizeKeySet the key set becomes available for key set rolling with RollKeySet. Only the initial AKS (that is, at application creation) can have key set version 0x00.
Key set type	Key set	The key set type is defined in Section 7.8.2. A key set has only one type that means that all keys within a key set share the same type.

Application level properties and key set level properties are also shown in Figure 12.

At creation of an application, the size needed to store application keys is computed based on the number of key sets, the number of keys and the maximum key size, as defined in Section 7.8.7. This size is fixed, even though some key set of the application would not hold keys with maximum key size at some point in time.

7.8.5.1.3 Key Set Personalization

The personalization of a key set is done through the following steps.

Initialize key set: First the key set is initialized with InitializeKeySet specified in Section 7.8.6.3 initializing all keys of the key set with the values of the corresponding keys in the AKS, padded or truncated if necessary according to the given key type.

Change keys: Then the keys of the key set are updated with their key values and versions with ChangeKeyEV2 as defined in Section 7.8.6.2. Changing the keys of the AKS is allowed too.

Finalize key set: Finally once all keys have been set the key set is finalized with FinalizeKeySet specified in Section 7.8.6.4. Note that, however not recommended, it is still possible to change keys after the key set has been finalized.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantas IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-222c-405c-90fe-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

At application creation, all key sets are already initialized according to the key type given for the initial AKS with CreateApplication or CreateDelegatedApplication. The initial AKS is also finalized setting the key set version to AKSVersion given in the command, while the other key sets remain in the initialized state with key set version 0x00.

7.8.5.1.4 Key Set Rolling

Once one or more key sets beside the AKS have been finalized within an application. The AKS can then be updated to point to another key set through a RollKeySet. This action is called key set rolling. Rolling to a key set is restricted and has to fulfill some rules specified as following.

Key set version rule: The key set version of the key set targeted by RollKeySet has to be strictly bigger than the key set version of the AKS. Therefore it is also not possible to roll into key sets before they are finalized as these will always have key set version 0x00.

Key set type rule: The key set targeted by RollKeySet has to have a key type stronger or equal than the AKS key type. The strength of a key type is specified in Section 7.8.2.

If the preceding rules are fulfilled, the key set rolling can take place. The AKS is updated to point to the targeted key set. The key set numbering of all key sets is changed relatively to the new AKS that receives key set number 0.

The current authentication status is invalidated moving the PICC into VCState.DFNotAuthenticated. Any ongoing transaction is aborted as specified in Section 7.14.2.2.

Previous AKS and all key sets with key set version smaller than the new AKS are considered to be invalid targets for RollKeySet because they do not fulfill the key set version rule.

7.8.5.1.5 Key Set Life Cycle

The life cycle of a key set is shown on Figure 13. This life cycle shows both key set personalization, see Section 7.8.5.1.3, and key set rolling, see Section 7.8.5.1.4.

Note that the figure shows the life cycle of a single key set within the application. On a successful RollKeySet, not only the targeted key set is affected (moving to the Active state), but all key sets, as the key set number changes. This is shown in the figure by the arrows with the condition *this.KSN! = KeySetNo*, that is, the given *KeySetNo* is different from the key set number of the key set which state is shown in the diagram (*this.KSN*).

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Actions

ActInit:
Set this.KSV to 0x00.
Set this.KST to given KeySetType.
Initialize all keys with AKS key values.

ActChangeKey:
Update targeted key with given value and version..

ActFinal:
Set this.KSV to given KeySetVersion.

ActRoll:
Set this.KSN to (this.KSN - KeySetNo) mod KeySetCount

Legend

this.KSN: key set number of the shown key set

this.KSV: key set version of the shown key set

this.KST: key set type of the shown key set

AKS.KSN: key set number of current AKS

AKS.KSV: key set version of current AKS

AKS.KST: key set type of current AKS

target.KSV: key set version of key set targeted by given KeySetNo

target.KST: key set type of key set targeted by given KeySetNo

KeySetCount: the number of key sets within this application

Note: In case of failure, an error return code is returned and the shown key set remains in the same state. This is not shown explicitly.

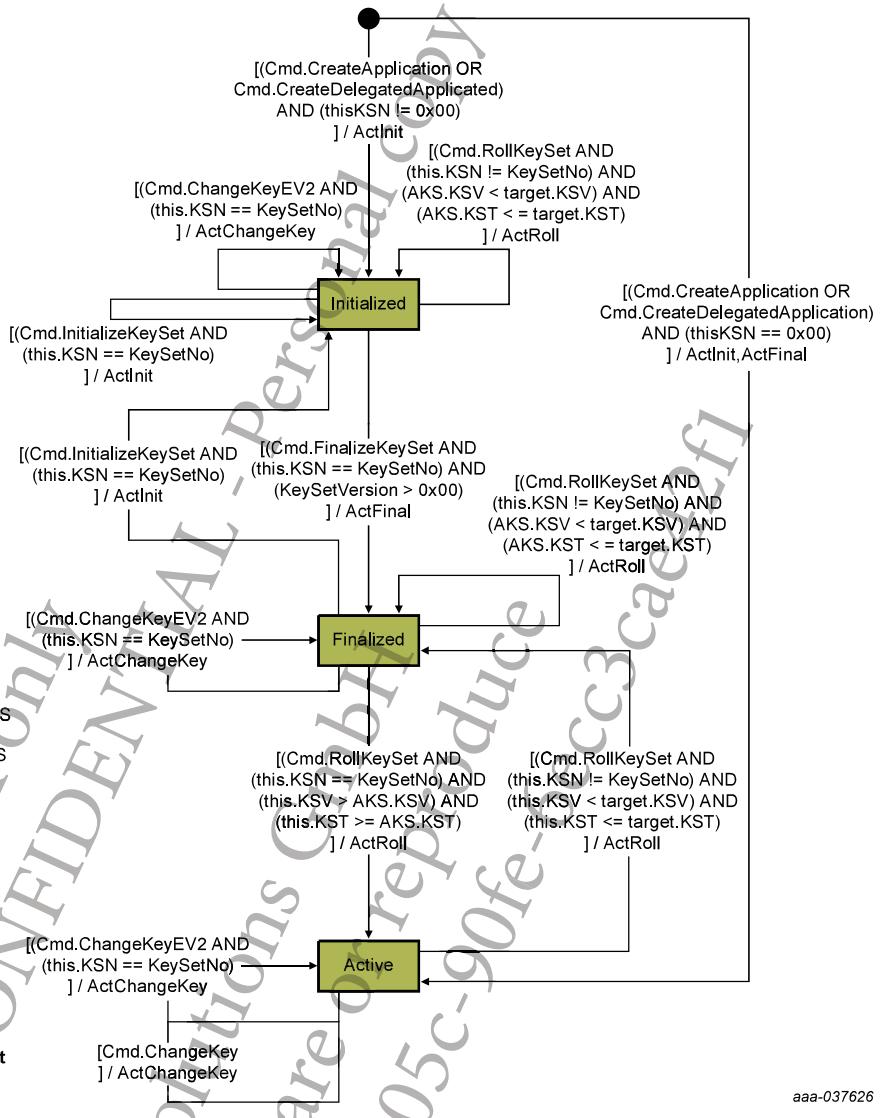


Figure 13. Application key set life cycle state diagram

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salatan
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-ecc3cae2f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.8.5.2 Application Keys

Application keys are defined in Table 35 and the following subsections. They are used to manage the security of an individual application.

Table 35. Keys at application level

Key Identifier	Symbol	Key number	Key Type	Change Key	Auth
Addressable keys:					
KeyID.AppMasterKey	$K_{AppMaster}$	0x00, targeting AKS	KeyType.AES128 KeyType.AES256	KeyID.AppMasterKey or none	yes
KeyID.AppRollKey	$K_{AppRoll}$	see Table 36, targeting AKS	KeyType.AES128 KeyType.AES256	KeyID.AppMasterKey or none	yes
KeyID.AppChangeKey	$K_{AppChange}$	see Table 36, targeting AKS	KeyType.AES128 KeyType.AES256	KeyID.AppMasterKey or none	yes
KeyID.AppUIDENCKey	$K_{AppUIDENC}$	See Set Configuration Option 0x04	KeyType.AES128 KeyType.AES256	KeyID.AppChangeKey or key itself	yes
KeyID.AppKey	K_{App}	0x00..0x0D, targeting any key set	KeyType.AES128 KeyType.AES256	KeyID.AppChangeKey or key from AKS with same key number or none	yes
KeyID.AppCommitReaderIDKey	$K_{AppCommitReaderID}$	see Table 40, targeting AKS	KeyType.AES128 KeyType.AES256	KeyID.AppChangeKey or key itself or none	yes
KeyID.VCProximityKey	$K_{VCProximity}$	0x21	KeyType.AES128 KeyType.AES256	KeyID.AppMasterKey	no
Reserved		0x22 … 0x23			
Non-addressable keys					
KeyID.AppTransactionMACKey	$K_{AppTransactionMAC}$	N/A	KeyType.AES128 KeyType.AES256	N/A	no
KeyID.AppDAMDefaultKey	$K_{AppDAMDefault}$	N/A	N/A	N/A	no

The KeyID.AppDAMDefaultKey and KeyID.AppTransactionMACKey are not held in the AKS or any key set of the application. They cannot be addressed by authentication commands. However, as they still are application level keys, they are listed and discussed in Table 35 for consistency reason. Also the KeyID.VCProximityKey is not part of the key sets.

With regards to authentication, the KeyID.AppMasterKey, KeyID.AppRollKey, KeyID.AppChangeKey, and KeyID.AppCommitReaderIDKey are part of the AKS. This means that if a roll key set is performed on the application, for authentication, the key values of the new AKS are to be used.

A key of the AKS can have multiple identities listed in Table 35. All keys of the AKS and other key sets are KeyID.AppKey and they can be used as defined in Section 7.8.5.2.7. Then a KeyID.AppKey of the AKS can have other identities: KeyID.AppMasterKey, KeyID.AppRollKey, KeyID.AppChangeKey, and

Provided under NDA only
CONFIDENTIAL - Personal copy for:
Seetii Sadata
Quantum Solutions GmbH
Do not copy, share or reproduce
5fc677d1-c2c-405c-90fe-6ec3cae4af1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

KeyID.AppCommitReaderIDKey. In case of overlapping identities, the priority on conflicting requirements is according to the order in the table.

7.8.5.2.1 AppMasterKey

The KeyID.AppMasterKey always has key number 0 of the AKS. KeyID.AppMasterKey is used to give access through successful authentication to

- file creation/deletion
- key updates
- application directory

depending on the application settings defined in Table 36.

Key type: The KeyID.AppMasterKey can be a KeyType.AES128 or KeyType.AES256 key depending on the current AKS key type.

Key change: The KeyID.AppMasterKey is changeable with ChangeKey or ChangeKeyEV2 with an active authentication with KeyID.AppMasterKey. KeyID.AppMasterKey can be frozen depending on AppKeySettings[b 0] defined in Table 36.

Authentication: The KeyID.AppMasterKey is available for authentication. As explained in Section 7.6.1, KeyID.AppMasterKey access rights can also be obtained through an ECC-based authentication with a KeyID.CARootKey and eventually reader certificate granting the access right.

7.8.5.2.2 AppMasterTmpKey

The KeyID.AppMasterTmpKey is a temporary symmetric to allow initial personalization and trust establishment in ECC-only applications, see Section 7.9.5.3. It can be addressed at key number 0 and has equivalent access rights as the KeyID.AppMasterKey.

Key type: The KeyID.AppMasterTmpKey can be a KeyType.AES128 or KeyType.AES256 key as defined during application creation.

Key change: The KeyID.AppMasterTmpKey is not changeable with ChangeKey or ChangeKeyEV2. For conventional applications, its value is derived from the KeyID.PICCAppDefaultKey at the time of authentication. For delegated applications, its value is fixed to the KeyID.AppDAMDefaultKey at application creation. ChangeKey or ChangeKeyEV2 can be used to disable the KeyID.AppMasterTmpKey.

Authentication: The KeyID.AppMasterTmpKey is available for authentication.

7.8.5.2.3 AppRollKey

The KeyID.AppRollKey is identified by the key number specified in AppKeySetSettings[b 3..0], see Section 7.8.5.3. KeyID.AppRollKey is used to allow through successful authentication the use of RollKeySet.

Key type: The KeyID.AppRollKey can be a KeyType.AES128 or KeyType.AES256 key depending on the current AKS key type.

Key change: If not frozen, depending on AppKeySettings[b 7..4] defined in Table 36, the value and version of KeyID.AppRollKey are changeable with ChangeKey or ChangeKeyEV2 with an active authentication with KeyID.AppMasterKey.

Key change: If not frozen, depending on AppKeySettings[b 7..4] defined in Table 36, the value and version of KeyID.AppRollKey are changeable with ChangeKey or ChangeKeyEV2 with an active authentication with KeyID.AppMasterKey.

Note that the definition of the KeyID.AppRollKey number is fixed at application creation by the AppKeySetSettings. It cannot be changed afterward.

Provided under NDA only. CONFIDENTIAL - Personal copy for: Sergi Santa Quintana. Do not copy/share/reproduce. 5107779c-c22-4050-0fc-66cc3cae4af1

Authentication: The KeyID.AppRollKey is available for authentication. As explained in Section 7.6.1, KeyID.AppRollKey access rights can also be obtained through an ECC-based authentication with a KeyID.CARootKey as defined by AppKeySetSettings.

7.8.5.2.4 AppChangeKey

The KeyID.AppChangeKey is identified by the key number specified in AppKeySettings[b 7..4], see Section 7.8.5.3 to be the change key. It is as well possible to have no KeyID.AppChangeKey if the KeyID.AppKeys are frozen or if an authentication with the KeyID.AppKey itself or its equivalent in the AKS is required.KeyID.AppChangeKey is used to allow through successful authentication the use at application level of InitializeKeySet, FinalizeKeySet, ChangeKeyEV2 and ChangeKey.

Key type: The KeyID.AppChangeKey can be a KeyType.AES128 or KeyType.AES256 key depending on the current AKS key type.

Key change: The value and version of KeyID.AppChangeKey are changeable with ChangeKey or ChangeKeyEV2 with an active authentication with KeyID.AppMasterKey.

Note that the KeyID.AppChangeKey number, that is, which key in the AKS (if any) is used as the KeyID.AppChangeKey, is configured via the AppKeySettings, see Section 7.8.5.3.

Authentication: The KeyID.AppChangeKey is available for authentication. As explained in Section 7.6.1, KeyID.AppChangeKey access rights can also be obtained through an ECC-based authentication with a KeyID.CARootKey and eventually reader certificate granting the access right defined by AppKeySetSettings.

7.8.5.2.5 AppCommitReaderIDKey

The KeyID.AppCommitReaderIDKey is part of the AKS. Its use for the Transaction MAC feature is described in Section 7.14.3. Its key number is specified in Section 7.12.1.5.1.

Key type: The KeyID.AppCommitReaderIDKey can be a KeyType.AES128 or KeyType.AES256 key depending on the current AKS key type.

Key change: The value and version of the KeyID.AppCommitReaderIDKey is managed as any other KeyID.AppKey of the AKS. Note that the number of the KeyID.AppCommitReaderIDKey is configured via the FileType.TransactionMAC file's access rights, see Section 7.12.1.5.1.

Authentication: The KeyID.AppCommitReaderIDKey is available for authentication. As explained in Section 7.6.1, KeyID.AppCommitReaderIDKey access rights can also be obtained through an ECC-based authentication with a KeyID.CARootKey and eventually reader certificate granting the required access right defined in Section 7.12.1.5.1.

7.8.5.2.6 AppUIDENCKey

The KeyID.AppUIDENCKey is a special role for one of the KeyID.AppKeys, as configured with SetConfiguration Option 0x04 when enabling fast confidential UID retrieval. Its usage is defined in Section 7.7.1.3.

Key type: The KeyID.AppUIDENCKey can be a KeyType.AES128 or KeyType.AES256 key depending on the current AKS key type.

Key change: The KeyID.AppUIDENCKey key change is no different than any regular KeyID.AppKey.

Authentication: As any other KeyID.AppKey, the KeyID.AppUIDENCKey remains available for authentication.

7.8.5.2.7 AppKey

An KeyID.AppKey is any key of the AKS or any other key set. The number of application keys are defined at application creation. Note that in generic descriptions like command tables the addressable range is defined by

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-022c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

the maximum number of application keys possible, that is, 14 or 0x0..0xD. NO_SUCH_KEY is returned if the targeted key is not supported by the currently targeted application. KeyID.AppKeys of the AKS are used to give access through successful authentication to, for example, file access operations as defined in Section 6.7.2.

Key type: An KeyID.AppKey can be a KeyType.AES128 or KeyType.AES256 key depending on the current AKS key type.

Key change: If not frozen, depending on AppKeySettings [b 7..4] defined in Table 36, KeyID.AppKey is changeable with ChangeKey or ChangeKeyEV2 with an active authentication with KeyID.AppChangeKey or with the targeted KeyID.AppKey (or the related KeyID.AppKey of the AKS if targeting a key outside the AKS), as defined in Table 36. Note that this setting can be overruled if this KeyID.AppKey has another identity (KeyID.AppMasterKey, KeyID.AppChangeKey or KeyID.AppRollKey).

Authentication: An KeyID.AppKey is available for authentication.

7.8.5.2.8 AppTransactionMACKey

The KeyID.AppTransactionMACKey is not part of the AKS. Its use for the Transaction MAC feature is described in Section 7.14.3.

Key type: The KeyID.AppTransactionMACKey is always a KeyType.AES key.

Key change: The KeyID.AppTransactionMACKey is changeable through a FileType.TransactionMAC file deletion/creation. According to the AppKeySettings[b 2], this will require an active authentication with the KeyID.AppMasterKey.

Authentication: The KeyID.AppTransactionMACKey is not available for authentication.

7.8.5.2.9 VCProximityKey

An application can set its own KeyID.VCProximityKey to support Proximity Check.

When this key override is enabled at application creation with CreateDelegatedApplication, the KeyID.VCProximityKey reuses the same addresses (0x21) as at PICC level.

Key Type: At application level, the KeyID.VCProximityKey can be a KeyType.AES128 or KeyType.AES256 key. The maximal key size is defined by the MaxKeySize parameter during application creation. In the absence of key sets, the key type is fixed by the application key type set with KeySett2. The initial key type will be according to KeySett2, that is, equivalent with the AKS. If both KeyType.AES128 and KeyType.AES256 are supported for an application, the actual key type can be changed with ChangeKey or ChangeKeyEV2.

Key Change: At application level, KeyID.VCProximityKey is immediately enabled (getting the value and version defined by the KeyID.PICCAppDefaultKey (for conventional applications) or KeyID.AppDAMDefaultKey (for delegated applications)) and require authentication with the KeyID.AppMasterKey for change.

Authentication: At application level, the KeyID.VCProximityKey is also not available for authentication.

7.8.5.2.10 AppDAMDefaultKey

The KeyID.AppDAMDefaultKey is not part of the AKS but part of the application. This key is used on delegated application creation instead of the KeyID.PICCAppDefaultKey. Its use is defined in Section 7.11.2.2.

Key type: The KeyID.AppDAMDefaultKey has no type associated to it. Instead it consists of a 24-byte or 32-byte string for its value and a 1-byte key version. In case an applications supports KeyType.AES256, as defined by MaxKeySize or Keysett2 during application creation, a 32-byte string must be provided. If not, a 24-byte string is provided for compatibility reasons with MIFARE DESFire.

On key initialization, the key value used depends on the key type of the targeted application. For KeyType.AES128, only the left most 16 bytes are used.

Provided under NDA only. COMPANY CONFIDENTIAL - Personal copy for: Sergio Salinas Quintanilla, Dialog Semiconductor GmbH. Do not copy, share or reproduce. 5f6b7b9d-c22c-4b3c-b0f6-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Key change: The KeyID.AppDAMDefaultKey is not changeable as it is not used anymore after application creation.

Authentication: The KeyID.AppDAMDefaultKey is not available for authentication.

7.8.5.3 Application Key Settings

The application key settings are defined in two bytes as is specified in Table 36 for generic application key settings, AppKeySettings, and Table 37 for application key set settings, AppKeySetSettings.

GetKeySettings allows retrieving both AppKeySettings and AppKeySetSettings if the application has been created with more than one key set (see CreateApplication or CreateDelegatedApplication). Otherwise only AppKeySetSettings of Table 36 are returned.

ChangeKeySettings allows updating the AppKeySettings of Table 36. The AppKeySetSettings defining the KeyID.AppRollKey is set at application creation with CreateApplication or CreateDelegatedApplication and fixed thereafter.

Table 36. AppKeySettings byte definition

AppKeySettings	Bit Index	Description	Value
Change key access right	Bit 7 - 4	Access condition for Change Key and ChangeKeyEV2 targeting an KeyID.AppKey	0x0..0xD: active authentication with specified key, that is, KeyID.AppChangeKey (if existing in AKS), or equivalent access rights granted through asymmetric authentication.
			0xD: [Optional, if enabled for the application, see Section 7.6.1] free access over I^2C for ChangeKey, InitializeKeySet and FinalizeKeySet. Over NFC, authentication is required.
			0xE: active authentication with targeted KeyID.App Key if within AKS, or its equivalent with the same key number within AKS, if targeting key outside AKS. Authentication with KeyID.AppMasterKey is required for InitializeKeySet and FinalizeKeySet.
			0xF: no access. All keys of all key sets beside the KeyID.AppMasterKey are frozen.
AppKeySettings changeable	Bit 3	AppKeySettings is	0: frozen
			1: changeable with KeyID.AppMasterKey authentication
File create/delete configuration	Bit 2	KeyID.AppMasterKey authentication is	0: required for access to create file commands and DeleteFile command.
			1: not required for access to create file commands and DeleteFile command.
File directory access configuration	Bit 1	KeyID.AppMasterKey authentication is	0: required for access to GetFileIDs, GetISOFileIDs, GetFileSettings, and GetKeySettings commands.
			1: not required for access to GetFileIDs, Get ISOFileIDs, GetFileSettings, and GetKeySettings commands.
KeyID.AppMasterKey changeable	Bit 0	KeyID.AppMasterKey is	0: frozen
			1: changeable with KeyID.AppMasterKey authentication

Provided under NDA only - Personal copy for: COMPANY CONFIDENTIAL Sergii Salata Quantas IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-4055-90fe-6e8c5e42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

If not frozen, the KeyID.AppMasterKey, KeyID.AppRollKey or KeyID.AppChangeKey can only be changed with an authentication with the KeyID.AppMasterKey independently of the AppKeySettings[b 7..4] from Table 36.

Table 37. AppKeySetSettings byte definition

AppKeySetSettings	Bit Index	Description	Value
RFU	Bit 7-4		RFU
Roll key access right	Bit 3-0	Access condition for RollKey Set	0x0..0xD: active authentication with specified key, that is, KeyID.AppRollKey (if existing in AKS), or equivalent access rights granted through asymmetric authentication.
			0xD: [Optional, if enabled for the application, see Section 7.6.1] free access over I^2C for RollKeySet. Over NFC, authentication remains required.
			0xE..0xF: RFU

As a reminder KeyID.AppMasterKey, KeyID.AppRollKey and KeyID.AppChangeKey are part of the AKS but are used to manage as well keys for other key sets of the same application.

If KeyID.AppChangeKey is set to require authentication of the KeyID.AppKey targeted, this means authentication with the key at the same key number of the AKS is required if targeting keys of other key set with ChangeKeyEV2.

Note that these access conditions can also be obtained via equivalent access rights obtained via ECC-based asymmetric authentication, see Section 7.5.6 and Section 7.6.

7.8.6 Key Management Commands

This section gives the overall description of the key management commands as most of them apply to both PICC and application level.

7.8.6.1 ChangeKey Command

Both at PICC and application level, changing keys is possible with the command ChangeKey as defined in Section 8.4.1 and Figure 36.

At PICC level, the command is rejected when targeting the KeyID.PICCMasterKey if this is disallowed by the PICCKeySettings, see Section 7.8.4.2.

At application level, the command is rejected if the targeted key is frozen as defined in the AppKeySettings, see Section 7.8.5.3. If the SAI is set and no second application is selected, the command is rejected.

The command is also rejected at application level when targeting an application specific KeyID.VCProximityKey and trying to change the key type into KeyType.AES256, while this is not supported by the application key type (as defined by MaxKeySize or KeySett2 during application creation).

The command is also rejected if there is no active authentication with the relevant change key. For the PICC level, this is defined in Table 32.

For the application level, KeyID.AppMasterKey, KeyID.AppChangeKey and KeyID.AppRollKey require authentication with KeyID.AppMasterKey. Also if present, the application specific KeyID.VCProximityKey requires authentication with KeyID.AppMasterKey. All other KeyID.AppKeys require authentication withKeyID.AppChangeKey, according to the change key setting in Table 36.

The required access rights can also be achieved through asymmetric authentication, see Section 7.6. If KeyID.AppChangeKey is configured to 0xE, this means that the CertAccessRights must include the access

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sascia Sata
Contactless solutions GmbH
Do not copy, share or reproduce
5fc07e91-022-405c-90fe-6ecc3cae4ff1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

condition equivalent to the targeted key. For ECC-only applications, the command can be used to disable the temporary KeyID.AppMasterTmpKey. For further details, see Section 7.9.5.3.

Under EV2 Secure Messaging, i.e. if in VCState.DFAuthenticatedAES or in VCState.AuthenticatedECC, the secure messaging is as applied under CommMode.Full, see Section 7.5.10.8. For the plaintext, two cases can be distinguished:

Targeted key equal to authenticated key: If the targeted key is equal to the authenticated key (that is, *KeyNo == getKeyNo(*AuthKey*)*), the plaintext is constructed as follows:

- KeyType.AES128: Key Data = *NewKey* || *KeyVer,* (16 bytes + 1 byte)
- KeyType.AES256: Key Data = *NewKey* || *KeyVer,* (32 bytes + 1 byte)

NewKey is the new key. *KeyVer* is the related key version. *KeyVer* is only present in case of targeting KeyType.AES128; for the other key types the key version is included in *NewKey*. The normal EV2 secure messaging for CommMode.Full is applied on the command. The response is sent in plain, as the authentication is lost (see below).

Note that in VCState.AuthenticatedECC, this case cannot occur.

Targeted key different from authenticated key: If the targeted key is not equal to the authenticated key (i.e. KeyNo! = getKeyNo(AuthKey)), the plaintext is constructed as follows:

- KeyType.AES128: Key Data = (*NewKey*⊕*OldKey)* || *KeyVer* || *CRC32NK,* (16 + 1 + 4 bytes)
- KeyType.AES256: Key Data = (*NewKey*⊕*OldKey)* || *KeyVer* || *CRC32NK,* (32 + 1 + 4 bytes)

NewKey is the new key value, while *OldKey* is the old key value currently present in the targeted key entry.

If the key types of the *NewKey* and *OldKey* differ, the *OldKey* is truncated or padded with zeros to match the target key type size. *KeyVer* is the new version.

The *CRC32NK* is the 4-byte CRC value computed over *NewKey*. The CRC is computed according to IEEE Std 802.3-2008 (FCS Field) [32].

The normal EV2 secure messaging for CommMode.Full is applied on both the command and the response.

Note that in VCState.AuthenticatedECC, this case always applies. In VCState.DFAuthenticatedAES, this case applies always if not targeting KeyID.AppMasterKey.

If free access over I^2C is enabled for the selected application, see Section 7.6.1, and 0xD is configured as the change key access condition in AppKeySettings, the command can be executed in VCState.NotAuthenticated. In that state, no secure messaging is applied.

The key value (*NewKey*) and the related key version (*KeyVer*) retrieved after decryption is used to change the targeted key. If the length does not match with the targeted key type, the command is rejected.

If targeting application keys, only keys of the AKS can be changed. If targeting the KeyID.PICCMasterKey, the key type is updated with the type specified in KeyNo[b 7..6]. If the key type changes, this also affects the KeyID.PICCDAMKeys. Therefore, they are disabled in this case.

At the PICC level, if a currently disabled key is targeted, the key gets enabled and can from then on be used. Note that in that case the *OldKey* value used for ChangeKey is always all zero (independently of whether the key has been initialized before).

If the key used for current active authentication AuthKey is changed, then the authentication is invalidated. The PICC moves into VCState.NotAuthenticated.

Provided under NDA only - Personal copy for: COMPANY CONFIDENTIAL Sebastian Schmitz Quantas IT Solutions GmbH Do not copy, share or reproduce 5fc07794-224-4056-90fe-6ec3cae42f1

7.8.6.2 ChangeKeyEV2 Command

Changing keys is also possible with the command ChangeKeyEV2 as defined in Section 8.4.2 and Figure 37.

It includes an additional parameter *KeySetNo* to identify the key set to update key from. The addressing is done relatively to the AKS. Note that if *KeySetNo* is different from zero, always the case where targeted key is different from authenticated key needs to be applied. All secure messaging variants apply, as described for ChangeKey in Section 7.8.6.1, but for this command *KeySetNo* shall be included in the integrity protection, that is, *Cmd* || *KeyNo* is replaced by *Cmd* || *KeySetNo* || *KeyNo*.

If the targeted key set does not exist, the command is rejected. Note that at PICC level, only *KeySetNo* set to 0x00 is accepted, as there are no key sets at PICC level. The same is true at application level, when targeting keys that are not regular KeyID.AppKeys, i.e. with a *KeyNo* outside 0x0..0xD.

For the rest, behavior of ChangeKey and ChangeKeyEV2 is identical. In other words, ChangeKey behaves as ChangeKeyEV2 with *KeySetNo* set to 0x00.

7.8.6.3 InitializeKeySet Command

Initializing a key set is possible with the command InitializeKeySet as defined in Section 8.4.3 and Figure 38.

This command initializes the key set with *KeySetNo* as following:

- sets the type of the key set to the given *KeySetType*
- sets the key set version to 0x00
- sets the content of all keys of the key set to the value and version of the corresponding key in the AKS. If required due to *KeySetType* of different length, the corresponding key is padded with zero bytes or truncated to the targeted length. This means that when initializing with KeyType.AES256, while the current AKS is of KeyType.AES128, 16 zero bytes are added.

If the PICC level has been selected, then InitializeKeySet is rejected. If the SAI is set and no second application is selected, the command is rejected.

The InitializeKeySet requires active authentication with KeyID.AppChangeKey or KeyID.AppMasterKey according to the AppKeySettings, see Table 36.

If *KeySetNo* is 0x00, the command is rejected as it is forbidden to initialize the AKS. A *KeySetNo* bigger than the number of key set in the application is also not accepted.

If the targeted KeySetType is not supported by the application, the command is rejected. This happens if targeting KeyType.AES256, while the MaxKeySize set during application creation was only 16 bytes.

The command InitializeKeySet requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.8.6.4 FinalizeKeySet Command

Finalizing a key set is possible with the command FinalizeKeySet as defined in Section 8.4.4 and Figure 39

This command finalizes the key set targeted by *KeySetNo* by setting the key set version to the given *KeySetVersion*. The key set version value is chosen by the application owner and should be determined in accordance to the rules specified in Section 7.8.5.1.4 to make it a valid candidate for key set rolling. But no comparison against the current AKS is done at this step. The only value that is rejected at this point is 0x00. Fulfilling rules specified in Section 7.8.5.1.4 is mandatory to be able to change the AKS to point to this key set with RollKeySet.

If the PICC level has been selected, then FinalizeKeySet is rejected. If the SAI is set and no second application is selected, the command is rejected.

Provided under NXP CONFIDENTIAL - Personal copy for: Sergio Santos, Quadrant Solutions GmbH
Do not copy, share or reproduce
5fc0776d-e2c-40bc-90fe-6e43cae42f1

The FinalizeKeySet requires active authentication with KeyID.AppChangeKey or KeyID.AppMasterKey according to the AppKeySettings, see Table 36.

If *KeySetNo* is 0x00, the command is rejected as it is forbidden to finalize the AKS. A *KeySetNo* bigger than the number of key set in the application is also not accepted.

The current version of the key set to be finalized has to be 0x00, i.e the key set must be in the initialized state, otherwise the command is rejected.

The command FinalizeKeySet requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.8.6.5 RollKeySet Command

Rolling to a key set is possible with the command RollKeySet as defined in Section 8.4.5 and Figure 40.

This command changes the AKS to point to the key set currently targeted with the given *KeySetNo*. After successful execution, the keys of the new AKS are from then on available for authentication. Any active authentication is canceled and the PICC is set to VCState.NotAuthenticated. Any ongoing transaction is aborted as if executing AbortTransaction.

If the PICC level is selected, RollKeySet is rejected. If the SAI is set and no second application is selected, the command is rejected.

Rules defined in Section 7.8.5.1.4 have to be fulfilled. Otherwise the command is rejected.

RollKeySet needs an active authentication with the KeyID.AppRollKey, as configured in the AppKeySetSettings, see Table 37.

If *KeySetNo* is 0x00 or bigger than the number of key set in the application, the command is not accepted.

The command RollKeySet requires CommMode.MAC. Information on authentication and secure messaging dependent structure of the command can be found in Section 7.5. As the authentication is lost, the response is sent with CommMode.Plain.

7.8.6.6 GetKeySettings Command

Retrieving key settings is possible with the command GetKeySettings as defined in Section 8.4.6 and Figure 41.

At PICC level, if Bit 1 of PICCKeySettings is not set, see Table 33, one of the following authentications is required:

- a symmetric authentication with the KeyID.PICCMasterKey or the KeyID.PICCDAMAuthKey
- an asymmetric authentication granting KeyID.PICCMasterKey access rights
- an asymmetric authentiation granting delegated application management access rights

The following values are returned:

- *KeySetting* is set to the current PICCKeySettings
- Bit 7-6 of *MaxNoOfKeys* represents the key type of the KeyID.PICCMasterKey, encoded as defined in Table 31. Other keys have their key type aligned with this one or fixed to KeyType.AES as described in Section 7.8.4. Bit 5-0 of *MaxNoOfKeys* is kept to 0x01 for backward-compatibility reasons as the number of PICC keys is not configurable by the user anyhow.

At application level, an authentication with the KeyID.AppMasterKey is required if Bit 1 of AppKeySettings is not set, see Table 36. The following values are returned:

- *KeySetting* is set to the current AppKeySettings

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergey Sakata
Quartet IT Solutions GmbH
Do not copy, share or reproduce
5fc07779d22c4955-90fe-6ec03cae4f4

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

- Bit 7-6 of *MaxNoOfKeys* represents the key type of the AKS, encoded as defined in Table 31. Bit 5-0 is set to the number of application keys, that is, per key set, see Section 7.8.5.1.2. If the application does not contain any keys, 0x00 will be returned.
- *AKSVersion* is an optional parameter that is only returned if the selected application has more than one key set. In this case, it represents the current key set version of the AKS.
- *NoKeySets* is an optional parameter that is only returned if the selected application has more than one key set. In this case, it represents the number of key sets within the application.
- *MaxKeySize* is an optional parameter that is only returned if the selected application has more than one key set. In this case, it represents the maximal key size of a key set within the application.
- *AppKeySetSett* is an optional parameter that is only returned if the selected application has more than one key set. In this case, it represents the current AppKeySetSettings, see Table 37.

If an Option is given, the metadata of a specific key group is returned.

Table 38. GetKeySettings Key Groups

Option	Key Group
0x01	KeyID.ECCPrivateKey
0x02	KeyID.CARootKey

If two applications are selected, the command is applied on the primary selected application.

Under active authentication, the command GetKeySettings requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.8.6.7 ChangeKeySettings Command

Changing key settings is possible with the command ChangeKeySettings as defined in Section 8.4.7 and Figure 42.

At PICC level, an authentication with the KeyID.PICCMasterKey is required. If Bit 3 of PICCKeySettings is not set, see Table 33, the PICCKeySettings are frozen and the command is rejected.

At application level, an authentication with the KeyID.AppMasterKey is required. If Bit 3 of AppKeySettings is not set, see Table 36, the AppKeySettings are frozen and the command is rejected.

The command consists of one parameter *KeySetting* which is used to configure the PICCKeySettings at PICC level, or the AppKeySettings of the currently selected application.

If two applications are selected, the command is applied on the first selected application.

The command ChangeKeySettings requires CommMode.Full. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.8.6.8 GetKeyVersion Command

Getting the key version of an addressable key or the key set versions of all key sets within the selected application is possible with the command GetKeyVersion as defined in Section 8.4.8 and Figure 43.

KeyNo and optionally *KeySetNo* indicate which information is requested. *KeyNo* Bit 7 (SAI) indicates which application is targeted if multiple are selected. If the SAI is set and no second application is selected, the command is rejected. Bit 6 indicates if *KeySetNo* parameter is present. When targeting key version retrieval, Bit 5-0 define the actual key number of the targeted key. If the key does not exist or is still disabled (only applicable for keys at PICC level), the command is rejected. Within an application, *KeySetNo* allows targeting keys in key sets different from the AKS. If present, it indicates number of the key set targeted (relatively to the AKS). *KeySetNo* can also indicate key set version retrieval via its Bit 7. For key set version retrieval, the command is

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sebastian Salama
Quantum ITS Solutions GmbH
Do not copy, share or reproduce
5f0f7f91-c22c-4052-90fe-fec5cae42f1

rejected if the PICC level is selected, as it does not contain key sets, or if the selected application does not hold more than one key set.

When retrieving a key version, a single-byte *KeyVer* is returned holding the key version. If targeting disabled keys, 0x00 is returned.

When retrieving key set versions, a string *KeySetVer* with length equal to the number of key sets within the selected application is returned. The key set versions are returned ordered by ascending key set number. This means that the key set version of AKS is the first byte returned.

This command can be issued without an active authentication, but if there is an active authentication the command GetKeyVersion requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.8.7 Memory Consumption

Memory allocation is done in 32-byte blocks, see Section 7.7.4.

The memory for PICC level keys and configuration is not reduced from the user memory.

The memory for application keys is allocated at application creation and can be computed as follows, depending on CreateApplication or CreateDelegatedApplication parameters:

- If the application does not hold any keys: 0 blocks. KeySett2 Bit 3-0 are set to 0x0.
- Else: *NoKeySets**⌈*(21+(NoAppKeys * (MaxKeySize + 4)))/32*⌉ blocks)

With the following definitions:

- *NoKeySets*: as set with NoKeySets. If *NoKeySets* is not present, it is set to 1.
- *MaxKeySize*: 16 or 32 byte
- *NoAppKeys*: as set with KeySett2 Bit 3-0

Additionally, an application allocates memory for the following item:

- Application-specific KeyID.VCProximityKey (*KeySett3*, Bit 1 set): 3 blocks
- Delegated applications enabling KeyID.AppMasterTmpKey: 2 blocks

This memory will not be released on application deletion or (for delegated applications) formatting. The memory can only be released by formatting the complete PICC, see Section 7.7.4.2.

7.9 Asymmetric Key management

This chapter covers the key management aspects of asymmetric keys for MF3E(H)x3. The key management of symmetric keys is detailed in Section 7.8.

MF3E(H)x3 distinguishes private and public keys and the way they are managed:

- KeyID.ECCPrivateKey: This is the private key of an asymmetric ECC key pair, which is used to authenticate the MF3E(H)x3 towards external parties. The management of these keys is detailed in Section 7.9.1.
- KeyID.CARootKey: This is the public key of an asymmetric ECC key pair, which is used to authenticate an external party towards the MF3E(H)x3. This key is written to the MF3E(H)x3 with ManageCARootKey. This is further detailed in Section 7.9.2.

Public keys can be exported with ExportKey. This is defined in Section 7.9.3. This command is used to export the actual public key value of a KeyID.CARootKey. KeyID.ECCPrivateKeys cannot be exported.

At PICC level, some predefined asymmetric key entries are available. This is defined in PICClevel. At application level, application-specific asymmetric key entries are to be created and managed. This is summarized in Section 7.9.5.

Provided under NDA
COMPANY CONFIDENTIAL - Personal copy for:
Siegfried Data
Quintet3 IT Solutions GmbH
Do not copy, share or reproduce
5fc077d-92c-405c90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.9.1 KeyID.ECCPrivateKey Management

7.9.1.1 ManageKeyPair command

The generation or import of KeyID.ECCPrivateKeys is possible with the command ManageKeyPair as defined in Section 8.5.1 and Figure 44.

MF3E(H)x3 supports up to five ECC Key Pairs per application. At the PICC level, two private keys are preprovisioned as defined in Section 7.8.4. The key pairs are associated with a specific curve via CurveID.

Each ECC key pair is assigned to a specific application area and potentially to a specific protocol via KeyPolicy. Typically, best security practice is to use each key for a single purpose. Therefore, if allowing multiple usages for the same key, the implication from security perspective must be assessed. For example, if allowing both generic ECC Sign operations and ECC-based Transaction Signature and/or Secure Dynamic Messaging with the same key, the generic ECC Sign operation may be abused to forge Transaction or SDM signatures.

Generation or import of a new key pair requires authentication granting KeyID.AppMasterKey, or at PICC-level KeyID.PICCMasterKey access rights, applying CommMode.Full. Key pair replacement requires write-access as specified with WriteAccess when the KeyID.ECCPrivateKey was created.

During key pair generation, the private key is securely stored on-card and the public key is returned to the caller. In the case of import, the private key is to provided via PrivateKey. In this case, the public key is not returned.

It is also possible to update only the metadata, that is, KeyPolicy and access rights, of an existing key entry. This will not affect the current key value. For metadata update, also WriteAccess as configured for the targeted key entry is required. In this case, the command is rejected if the CurveID is not set to the curve associated with the current key.

7.9.1.2 KeyID.ECCPrivateKey Key Usage Limit

To allow mitigating potential future attack scenarios, KeyID.ECCPrivateKeys can be configured with a key usage limitation. This allows limiting the amount of private key computations, and thus related trace collection for side-channel attacks. Next to attack mitigation, this feature can also be used to limit the usage of a card/device. Potentially, the limit can be increased in the field, for example, if the end user pays for additional service. The key usage limitation (KeyUsageCtrLimit) is configured through KUCLimit.

Once enabled for a particular KeyID.ECCPrivateKey, any private key usage will be counted through a KeyUsageCtr associated with that KeyID.ECCPrivateKey. This means that the KeyUsageCtr shall be incremented by one before the private key operation of the following operations:

- ISOInternalAuthenticate for Card-Unilateral Authentication, see Section 7.5.7.
- ReadData or ISOReadBinary when applying Secure Dynamic Messaging with ECDSA SDMSIG, i.e. only when SDMSIG is targeted to be read out, see Section 7.5.11.1.10.
- CryptoRequest with action 0x03 or VDE_ECDSASign for ECC signature generation, see Section 7.15 or Section 7.19.3.3.
- ISOGeneralAuthenticatePart2 or ISOGeneralAuthenticateFinal for ECC-based Mutual Authentication, see Section 7.5.6. Increment is only done for mutual authentication, as there are no private key operations executed by MF3E(H)x3 on Reader-Unilateral Authentication.
- CommitTransaction if the targeted application has enabled Transaction Signature, see Section 7.14.4.

Any updates to the KeyUsageCtr are written with antitearing protection, guaranteeing that the counter will in case of tearing either hold the previous or the targeted value.

If the configured KeyUsageCtrLimit has been reached, the related KeyID.ECCPrivateKey will be disabled. This means that the key cannot be used for private key computations, though the key entry can still be updated

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Sala, TTS Solutions GmbH
Guarantee IT Solutions GmbH
Do not copy, share, or reproduce
5fc07792-2240-4015-9bfe-6ec2cce4211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

(and potentially reenabled) if the required authentication to do so can still be gained. If the KeyUsageCtrLimit is disabled, private key operations are not counted.

When only updating metadata with ManageKeyPair, it is possible to disable or change the KeyUsageCtrLimit without affecting the current KeyUsageCtr value. Putting the limit to a value equal or lower than the current value, will immediately disable the key entry. When changing the current key value through an import or generate key pair action, the current KeyUsageCtr value shall be reset to zero.

It is also possible to freeze the current KeyUsageCtrLimit. This can be done through KeyPolicy Bit 15. Once the KeyUsageCtrLimit has been frozen, it cannot be updated anymore. This means that a ManageKeyPair updating metadata will be rejected if KUCLimit has a value different from the currently configured KeyUsageCtrLimit or if KeyPolicy Bit 15 is not set. It is still possible to change the limit configuration by generating or importing a new key pair.

Enabling the key usage limit feature may create a denial-of-service risk. For typical use cases, the risk should be limited if, for example, configuring a limit of one million, or if preceding authentication of the external party is required before the MF3E(H)x3 private key operation.

It is essential to properly protect the KeyID.ECCPrivateKey write-access, as the right to update the key entry also allows to update and/or disable the KeyUsageCtrLimit.

7.9.1.3 KeyID.ECCPrivateKey Information Retrieval

MF3E(H)x3 supports information retrieval with regards to KeyID.ECCPrivateKey by the following means:

- Retrieving metadata on the existing KeyID.ECCPrivateKeys. This can be done with GetKeySettings as defined in Section 7.8.6.6.

MF3E(H)x3 does not support exporting private keys or the related public keys. Note that the related public key is typically stored via a certificate in a FileType.StandardData file. If the certificate is not created at the time of KeyID.ECCPrivateKey generation or import, the public key may be temporarily stored in the file and later overwritten with the certificate. Note that this means one needs to be careful when generating a key pair ManageKeyPair and putting the WriteAccess condition to 0xF. If the public key in the response gets lost, one is not able to regenerate the key entry. Therefore, it is not recommended to put WriteAccess to 0xF before the public key has been received.

7.9.2 KeyID.CARootKey Management

7.9.2.1 ManageCARootKey command

The writing of KeyID.CARootKeys holding a concrete public key value is possible with the command ManageCARootKey as defined in Section 8.5.2 and Figure 45.

MF3E(H)x3 supports up to five KeyID.CARootKeys per application.

The public keys are associated with a specific curve via CurveID. Note that MF3E(H)x3 does not validate the provided public key.

Each KeyID.CARootKey has an associated set of access rights via AccessRights, which can be granted to the host after successful authentication depending on the presented certificates. Note that AccessRights is encoded LSB first.

All KeyID.CARootKeys can optionally be associated with a trusted issuer name via IssuerLen and Issuer. If a trusted issuer name is set, this is compared against the Issuer field of the public key certificate provided during the authentication. In the case of chaining, the parent certificate issuer must match. Note that the implementation stores the SHA-256 hash of the provided Issuer to allow for fixed memory consumption.

Provided under NXP COMPANY CONFIDENTIAL Personal copy for: Sergio Santa Cruz Guzman Do not copy, share or reproduce 5fc0a79d-c22c-405c-906e-6ec03cae4ff1

Creation of KeyID.CARootKeys requires authentication granting KeyID.AppMasterKey, or at PICC-level KeyID.PICCMasterKey, access rights, applying CommMode.Full. Updating an existing KeyID.CARootKey requires write-access as specified with WriteAccess when the entry was created. ReadAccess can be used to configure the communication mode and access condition for ExportKey if targeting this key. Certificate revocation, as defined in Certificate revocation, can be enabled via CRLFileCRLFileAID. This means that the file can potentially be located in another application. This is mainly to support use cases where the same KeyID.CARootKey is used across multiple applications, as it allows to share a single CRLFile. The command will be accepted if the targeted file does not exist or is not a CRLFile, but the certificate revocation will only be effectively enabled if a valid CRLFile is present.

7.9.2.2 KeyID.CARootKey Information Retrieval

MF3E(H)x3 supports information retrieval with regard to KeyID.CARootKey by the following means:

- Retrieving the public key value of the KeyID.CARootKey holding a public key value. This can be done with ExportKey as defined in Section 7.9.3.
- Retrieving metadata on the existing KeyID.CARootKeys. This can be done with GetKeySettings as defined in Section 7.8.6.6.

7.9.3 ExportKey command

The export of a public key value of a KeyID.CARootKey is possible with the command ExportKey as defined iin Section 8.5.3 and Figure 46.

The public key is exported in uncompressed point representation, that is, 0x04||*Pub.x*||*Pub.y*.

Whether authentication is required and what communication mode must be applied, depends on the ReadAccess configuration as set with ManageCARootKey.

7.9.4 PICC Level

7.9.4.1 KeyID.ECCPrivateKey entries

At PICC or MF level (AID 0x000000), in the default configuration, the MF3E(H)x3 is trust-provisioned during manufacturing with two key pairs from which the private keys are stored on the MF3E(H)x3 as KeyID.ECCPrivateKeys. For KeyPolicy and WriteAccess encoding, refer to ManageKeyPair API definition.

- PICC functionality key pair: (Priv.PICC,Pub.PICC)
- Unique key pair per die, that can be used to authenticate individual devices for executing PICC-level functionality, that is, operations that require KeyID.PICCMasterKey or KeyID.VCConfigurationKey access rights, or Delegated Application Management functionality.
- Priv.PICC is stored as KeyID.ECCPrivateKey KeyNo 0x00, with the following default configuration:
  - NISTP-256
  - KeyPolicy 0x0040, only allowing ECC-based Mutual Authentication. For privacy reasons, ECC-based Unilateral Authentication is not allowed by default for this key pair.
  - WriteAccess: 0x30, allowing replacement with ManageKeyPair after an authentication granting KeyID.PICCMasterKey access rights in CommMode.Full.
  - KUCLimit: disabled.
- Pub.PICC is trusted via the certificate Cert.PICC, as specified in Section 7.10.1.2 and Section 7.10.1.4.
  - OriginalityCheck key pair: see Section 7.21.1.1.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07790-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.9.4.2 KeyID.CARootKey entries

At PICC or MF level (AID 0x000000), in the default configuration, the MF3E(H)x3 is pre-allocated with two KeyID.CARootKey entries. By default, those KeyID.CARootKey entries are disabled, which means they cannot be addressed during the authentication. They can be enabled by writing a public key value with ManageCARootKey.

As these entries are pre-allocated, writing them does not affect the available user memory, as reflected by as reflected by FreeMem.

7.9.5 Application level

7.9.5.1 KeyID.ECCPrivateKey entries

At application level, MF3E(H)x3 supports up to five KeyID.ECCPrivateKey entries per application.

Application-level KeyID.ECCPrivateKeys can be assigned a special role. This is summarized in the following subsections.

7.9.5.1.1 KeyID.AppTransactionSIGKey

The KeyID.AppTransactionSIGKey is used for the Transaction Signature feature as described in Section 7.14.4. A KeyID.ECCPrivateKey is enabled as KeyID.AppTransactionSIGKey if referenced by the FileType.TransactionMAC file of an application. A KeyID.AppTransactionSIGKey must be enabled for Transaction Signature by having Bit 6 of its KeyPolicy set, see ManageKeyPair.

One must take care that this private key cannot be abused to forge Transaction Signature, therefore one must not set KeyPolicy Bit 4, as that would enable usage for generic ECDSA sign operations.

7.9.5.2 KeyID.CARootKey entries

At application level, MF3E(H)x3 supports up to five KeyID.CARootKey entries per application.

7.9.5.3 Initial trust in ECC-only applications

In general, MF3E(H)x3 supports applications with both symmetric and asymmetric keys. On application creation, which is introduced later in Section 8.6, the number of symmetric keys is defined. Asymmetric key entries (KeyID.ECCPrivateKey and KeyID.CARootKey entries) are only created after application creation via the commands introduced in this chapter. In a mixed application, authentication with the symmetric keys (typically the KeyID.AppMasterKey according to the default write-access rights for the asymmetric key management commands) ensures that a trusted secure channel is used for creating and provisioning those keys.

For ensuring initial trust in applications only holding asymmetric keys, a specific mechanism is foreseen to allow application creation and personalization in the field (for example, for delegated application management use cases). This mechanism allows creating an application with a temporary symmetric key, called KeyID.AppMasterTmpKey.This temporary key can be used for authentication with AuthenticateEV2First by addressing it via *KeyNo* 0x00. It has equivalent access rights to a regular symmetric KeyID.AppMasterKey. This allows doing application personalization steps, including the creation of KeyID.ECCPrivateKey with ManageKeyPair, and KeyID.CARootKey with ManageCARootKey. If already authenticated, it is also possible to renew the authentication with AuthenticateEV2NonFirst as for any other symmetric key.

The KeyID.AppMasterTmpKey can be enabled at application creation with CreateApplication or CreateDelegatedApplication via the *KeySett2*. As long as the KeyID.AppMasterTmpKey is enabled, this will be reflected by GetKeySettings.

Provided under NDA only
COMPANY CONFIDENTIAL
Security Solutions Group
Do not copy, share or reproduce
Original Document: 5fc07772-2c05-90fe-0ccc5ae4211

As soon as the application personalization is done, for example, a ManageCARootKey has been created, the KeyID.AppMasterTmpKey can be disabled via issuing a dummy ChangeKey or ChangeKeyEV2 targeting the KeyID.AppMasterTmpKey at *KeyNo* 0x00. Similar, as regular ChangeKey this will invalidate the current authentication with KeyID.AppMasterTmpKey. Once disabled no further authentication with AuthenticateEV2First is possible. Further personalization can then, if needed, be done using asymmetric authentication. To ensure as much as possible equivalent ChangeKey processing the following AppKeySettings are enforced during application creation:

- AppKeySettings. Bit 7-4 is 0x0 (Change key access condition).
- AppKeySettings. Bit 3 is '0' (AppKeySettings frozen).
- AppKeySettings. Bit 0 is '1' (KeyID.AppMasterKey changeable, allowing the command to be used for KeyID.AppMasterTmpKey disabling).

The KeyID.AppMasterTmpKey has the key value derived from the current KeyID.PICCAppDefaultKey, that is, derived at the time of authentication, for conventional applications see Section 7.11.1. For delegated applications see Section 7.11.2, it is set to the KeyID.AppDAMDefaultKey.

The different steps of a typical execution flow are detailed as follows:

1. Application creation
   - Execute CreateApplication or CreateDelegatedApplication enabling the KeyID.AppMasterTmpKey by setting KeySett2 Bit 3-0 to 0xF. This step may require preceding symmetric or asymmetric authentication at the PICC level, depending on the configuration.
2. Initial Application personalization
   - Select the newly created application with SelectApplication or ISOSelectFile.
   - Authenticate with the KeyID.AppMasterTmpKey using AuthenticateEV2First addressing KeyNo 0x00.
   - Create one or more KeyID.ECCPrivateKey entry with ManageKeyPair. Note that this step and the following are required to be able to do further personalization and/or operation in step 3 based on mutual asymmetric authentication.
   - Inject a card certificate signing the public key of the previously generated or imported key using CreateStdDataFile and WriteData.
   - Create a KeyID.CARootKey with ManageCARootKey.
   - If needed, depending on the use case, do further application personalization, like creating and initializing further files.
   - Disable the KeyID.AppMasterTmpKey by issuing a ChangeKey or ChangeKeyEV2 addressing KeyNo 0x00.
3. Further application personalization and/or operation
   - Once authentication with KeyID.AppMasterTmpKey was lost (for example, after power-on reset), reselect the application, if needed, and authenticate using ECC Unilateral or Mutual authentication with the newly created asymmetric key(s)
   - Further personalization or operation within the secure channel established via asymmetric authentication.

Many of the substeps of Step 2 are optional and may depend on the actual use case and needs of the application. For example, if the application is only intended for card unilateral authentication, one may only want to create a KeyID.ECCPrivateKey entry and some freely readable files. However, one needs to realize that after disabling the KeyID.AppMasterTmpKey there is no way to authenticate a reader anymore if no KeyID.CARootKey has been created.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.9.6 Memory Consumption

Memory allocation is done in 32-byte blocks, see Section 7.7.4.

The memory for asymmetric keys is allocated at their creation and is defined as follows:

- KeyID.ECCPrivateKey: 3 blocks.
- KeyID.CARootKey: 4blocks.

Additionally an application will allocate memory for the following item

- Delegated applications enabling KeyID.AppMasterTmpKey: 2 blocks

This memory can only be released when formatting the whole card with Format. When deleting an application or formatting a delegated application, this memory is not released.

7.10 Certificate management

MF3E(H)x3 distinguishes card and reader certificates.

Card certificates are used to authenticate the card or PICC during ECC-based mutual authentication, see Section 7.5.6, or as part of a card-unilateral authentication, see Section 7.5.7. Card certificates are further detailed in Section 7.10.1.

Reader certificates are used to authenticate the reader or PCD during ECC-based mutual or reader-unilateral authentication, see Section 7.5.6. Reader certificates are further detailed in Section 7.10.2.

Note: *This chapter refers to some concepts of application, file and data management, which may only become fully clear after having read the subsequent chapters.*

7.10.1 Card Certificates

7.10.1.1 Certificate Format and Storage

In general, MF3E(H)x3 is agnostic about the card certificate format and content of certificates stored by the user. In the default configuration, MF3E(H)x3 comes with two trust-provisioned (TP) certificates. These TP certificates are public-key certificates according to X.509 v3 format [7] as further detailed in Section 7.10.1.4.

MF3E(H)x3 supports storing card certificates in FileType.StandardData and FileType.BackupData files, as introduced in FileType.StandardData and FileType.BackupData. Certificates can be read and written using the supported data management (see Section 7.13.1) and standard ISO/IEC 7816-4 commands (see Section 7.17.3) for data file access. The regular file access rights as defined in Section 7.12.2 are applied.

If there is card-unilateral authentication, see Section 7.5.7, the certificate is often read upfront as part of the transaction. As in a typical use case, the certificate is freely readable, the potential impact on card-holder privacy must be assessed by the system integrator.

In the case of mutual authentication, the certificate is returned as part of the authentication command exchanges, see Section 7.5.6 and especially Section 7.5.6.5/Section 7.5.6.6.

The certificate file content must be formatted as shown in Figure 14, consisting of the following components:

CertificateLength(*CLEN***)** 3-byte unsigned integer, encoded LSB first. This integer defines the length of the subsequent certificate or certificate chain.

CertificateData The actual certificate or certificate chain of *CLEN* bytes. This data structure is the data that is returned during the ECC-based mutual authentication, that is, by *Msg.B.enc*in ISOGeneralAuthenticatePart2 or ISOGeneralAuthenticateFinal (see Section 7.5.6.5 or Section 7.5.6.6). It can consist of one or more X.509

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Certificate: NXP Semiconductors GmbH
Guardian: NXP Semiconductors GmbH
Demo copy, use or reproduce
5fcf79d-922-4bc-90fe-6ecc5ae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

certificates. Other formats can also be used because the MF3E(H)x3 and the mutual authentication protocol are agnostic about the card certificate format.

Padding Any data that further fills up the file. This data is ignored by the ECC-based mutual authentication.

The encoding with *CertificateLength*and *Padding* allows to pre-personalize MF3E(H)x3 with certificate files of a predefined length depending on the maximum certificate (chain) length, while the actual certificate content length may vary. This is, for example, the case for X.509 certificates due to the ECDSA signature encoding with ASN.1 DER INTEGERs where the length may differ for specific signatures. Also it allows to read out a certificate chain at once after having read the *CertificateLength*.

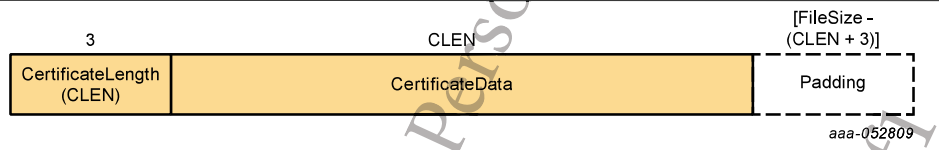


Figure 14. Certificate file content

For mutual authentication, the certificate is only returned after the reader has successfully authenticated. The reader obtains file access rights as encoded in the presented reader certificate, and the content of the targeted card certificate file is only returned if FileAR.Read or FileAR.ReadWrite is successfully granted.

MF3E(H)x3 supports card certificates at PICC level, which is further detailed in Section 7.10.1.2, and at application level, see Section 7.10.1.3.

7.10.1.2 PICC Level

At PICC or MF level (AID 0x000000), in the default configuration, the MF3E(H)x3 is trust-provisioned during manufacturing with two NXP certificates. The certificates are stored in the following two files:

- A FileType.StandardData file of 384 bytes with the following properties:
  - FileNo= 0x00; ISO File ID = 0xEF00.
  - FileAR.Read =0x3; FileAR.Write =0x0; FileAR.ReadWrite =0x0; FileAR.Change =0x0.
  - SecureDynamic Messaging and mirroring are not supported for this file.
  - Additional access rights are not supported for this file.
  - CommMode.Full
    This file holds the NXP PICC Certificate (Cert.PICC), as further defined in Section 7.10.1.4. At delivery, it will hold following content:
    - CertificateLength (3 bytes, LSB first): the actual length of the CertificateData.
    - CertificateData: the actual Cert.NXP.PICC.
    - Padding: remaining bytes in the file are set to all zero bytes.
- A file holding NXP Originality Certificate (Cert.Orig), as detailed in Section 7.21.1.2.

To allow personalization with customer-specific certificates at PICC level, these files can be updated. Also, additional FileType.StandardData files can be created with CreateStdDataFile, see Section 7.12.1.1. At PICC level, it is required to provide an ISO FileID. Secure Dynamic Messaging and additional access rights is not supported for FileType.StandardData files at the PICC level. The PICC level cannot hold other file types. The following file management commands, as introduced in Section 7.12, are also supported at PICC level: GetFileSettings, ChangeFileSettings, DeleteFile, GetFileIDs, GetISOFileIDs.

Provided by NXP Semiconductors GmbH
COMPANY CONFIDENTIAL - Personal copy for:
Setagaya Yuichi
Quantum Solutions
Do not share or reproduce
5fc0779d-2024-405c-9dfe-0ccc3ca8a21f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.10.1.3 Application Level

At the application or DF level, customized certificates can be stored in FileType.StandardData or FileType.BackupData files. The normal file and data management applies, as introduced in subsequent chapters. For usage during ECC-based mutual authentication, the file must be formatted as specified in Section 7.10.1.1.

7.10.1.4 Trust Provisioned Certificates

In the default configuration, the MF3E(H)x3 is trust-provisioned with two NXP certificates:

- Cert.PICC: the NXP PICC Certificate certificate. This is the certificate related to the (Priv.PICC, Pub.PICC), which can be used for PICC functionality as defined in Section 7.6.4.1. According to the definition below, the Cert.PICC has a maximum size of 322 bytes for 7-byte UID and 328 byte for 10-byte UID.
- Cert.Orig: the NXP Originality certificate. This is the certificate related to the (Priv.Orig,Pub.Orig), which can be used for originality checking as defined in Section 7.21.1. According to the below definition, the Cert.Orig has a maximum size of 301 bytes.

Both certificates are signed by NXP Trust Provisioning using dedicated key pairs:

- (Priv.CA.PICC, Pub.CA.PICC) is used for signing Cert.PICCs.
- (Priv.CA.Orig, Pub.CA.Orig) is used for signing Cert.Origs.

A TP certificate is a public-key certificate according to X.509 v3 format [8]. Optional fields from [8] have been omitted, that is, the TP certificate will not contain additional issuerUniqueID, subjectUniqueID, and extensions. The signature algorithm used is ECDSA with SHA-256.

A TP certificate is signed by NXP Trust Provisioning using a dedicated CA key pair for this product. The CA key pair can be retrieved from https://www.gp-ca.nxp.com/CA/getCA?caid=XXXXXXX filling in the CAID with the serialNumber encoded in the issuer name.

The issuer is set to

- Organizational name (O, OID 2.5.4.10): “NXP”
- Common name (CN, OID 2.5.4.3): for Cert.Orig, “NXP Orig RootCAvE2xx” with xx varying per product variant. For Cert.PICC, the issuer is set to “NXP PICC RootCAvE2xx”
- SerialNumber (OID 2.5.4.5): 14 digits encoding CAID

The subject contains a subset of GetVersion:

VendorID || HWMajorVersion || HWMinorVersion || SWType || SWSubType || SWMajorVersion || SWMinorVersion.

This is encoded in a description (OID 2.5.4.13) [7], using hexadecimal ASCII encoding.

For Cert.PICC, this version information is complemented with the 7-byte or 10-byte UID, encoded as a BIT STRING in a uniqueIdentifier (2.5.4.45) [7].

The two trust-provisioned NXP certificates are identifiable via following CA IDs:

- Cert.PICC: the NXP PICC Certificate certificate, using CA ID: 63709320121000
  - Certificates of this "NXP PICC RootCAvE2" can be retrieved via following link: https://www.gp-ca.nxp.com/CA/getCA?caid=63709320121000
- Cert.Orig: the NXP Originality certificate, using CA ID: 63709320111000
  - Certificates of this "NXP Orig RootCAvE2" can be retrieved via following link: https://www.gp-ca.nxp.com/CA/getCA?caid=63709320111000

Provided under NDA only - Personal copy for: COMPANY CONFIDENTIAL - Reproduce Qualitätssicherung Secure IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c022-405c-90fe-0cc2cae421f

7.10.2 Reader Certificates

Reader certificates are presented by the reader or PCD during ECC-based mutual or reader-unilateral authentication, see Section 7.5.6. They allow the MF3E(H)x3 to authenticate the reader before granting access to its assets.

The reader certificate format is detailed in Section 7.10.2.1. As the certificate is used to obtain access rights, it may hold a private extension encoding the access rights. If not, it is inherited from the parent certificate or targeted KeyID.CARootKey. This access right encoding is further detailed in Section 7.6.4.

MF3E(H)x3 accepts either a single certificate or a chain of up to two certificates. The certificate chain will be validated with the *CARootKey* addressed during the authentication. Individual certificates of up to 440 byte are supported. This means that in case of a certificate chain, Cert.A can be up to 880 byte.

7.10.2.1 Certificate Format

Reader certificates are public-key certificates according to X.509 v3 format [8]. They have the following specification, using ASN.1 distinguished encoding rules (DER) [14]. Optional fields from [8] have been omitted, that is, the MF3E(H)x3 does not support reader certificates containing a issuerUniqueID, a subjectUniqueID or extensions other than the one defined below. Other fields have been further restricted with regard to the supported length.

```
ReaderCertificate  ::= SEQUENCE {
    tbsReaderCertificate    TBSReaderCertificate,
    signatureAlgorithm      SignatureIdentifier,
    signatureValue    BIT STRING
}

TBSReaderCertificate  ::= SEQUENCE {
    version    [0] EXPLICIT INTEGER {v3(2)},
    serialNumber          INTEGER,
    signature             SignatureIdentifier,
    issuer                ReaderName,
    validity              Validity,
    subject         ReaderName,
    subjectPublicKeyInfo ECCNistP256PublicKey,
    extensions [3] EXPLICIT Extensions OPTIONAL
}

SignatureIdentifier  ::= SEQUENCE {
    algorithm OBJECT IDENTIFIER (ecdsa-with-SHA256)
}
ReaderName  ::= SEQUENCE OF ReaderDistinguishedName

Validity ::= SEQUENCE {
    notBefore    Time,
    notAfter     Time
}

Time  ::= CHOICE {
    utcTime     UTCTime,
    generalTime GeneralizedTime
}

ECCNistP256PublicKey  ::= SEQUENCE {
    algorithm     ECCNistP256AlgorithmIdentifier,
    subjectPublicKey    BIT STRING(SIZE(65))
}
CHOICE {
```

Provided under NDA only. COMPANY CONFIDENTIAL. Personal copy for: Sergi Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-405c-90fe-6ecc3cae42f1

```
    nistP256     ECCNistP256AlgorithmIdentifier,
    brainpoolP256r1    ECCbrainpoolP256r1AlgorithmIdentifier
}

ECCNistP256AlgorithmIdentifier  ::= SEQUENCE {
    eccOID     OBJECT IDENTIFIER (id-ecPublicKey),
    nistP256OID      OBJECT IDENTIFIER (secp256r1)
}

ECCbrainpoolP256r1AlgorithmIdentifier  ::= SEQUENCE {
    eccOID         OBJECT IDENTIFIER (id-ecPublicKey),
    brainpoolP256r1OID   OBJECT IDENTIFIER (brainpoolP256r1)
}

Extensions  - SEQUENCE SIZE (1..MAX) OF Extension

Extension  - SEQUENCE {
    extnID OBJECT IDENTIFIER,
    critical     BOOLEAN DEFAULT FALSE,
    extnValue    OCTET STRING
}

ARExtension  ::= SEQUENCE {
    arExtnId OBJECT IDENTIFIER (id-nxp-ar),
    critical BOOLEAN DEFAULT FALSE,
    arExtnValue    OCTET STRING
}

Ecdsa-Sig-Value  ::= SEQUENCE {
    r INTEGER,
    s INTEGER
}
```

version

The version shall be set to 0x02 reflecting the X.509 version 3 format.

serialNumber

MF3E(H)x3 supports serialNumbers of up to 4-byte INTEGERs for reader certificates.

signature/ signatureAlgorithm

MF3E(H)x3 supports the following signature algorithm: ECDSA with SHA256, reflected by OID 1.2.840.10045.4.3.2.

issuer

The issuer is set to a ReaderName as specified above. ReaderDistinguishedName is not further defined by this specification and not interpreted by MF3E(H)x3.

validity

MF3E(H)x3 ignores the validity of reader certificates.

subject

The subject is set to a ReaderName as specified above. ReaderDistinguishedName is not further defined by this specification and not interpreted by MF3E(H)x3.

Provided under NDA or CONFIDENTIAL Personal copy for: COMPANY Sergii Salata Quantum IT Solutions GmbH Do not copy, share or reproduce 5fc779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

subjectPublicKeyInfo

MF3E(H)x3 supports ECC key pairs over the following curves:

- CurveP-256, as defined by NIST in FIPS 186-4 [4].
- CurvebrainpoolP256r1, as defined in RFC 5639 [21]

This is reflected by OID 1.2.840.10045.2.1 for ECC. Additionally, the curve is encoded by and 1.2.840.10045.3.1.7 for NIST P-256 or OID 1.3.36.3.3.2.8.1.1.7 for brainpoolP256r1. The public keys are expected to be uncompressed point representation, that is, *0x04* || *Pub.ID.x* || *Pub.ID.y*.

extensions

MF3E(H)x3 supports extensions, as specified by [8].

MF3E(H)x3 recognizes and processes an optional private extension for access right encoding. This extension is reflected by an OID in the NXP range: 1.3.6.1.4.1.28137.64.1. The detailed access right encoding is given in Section 7.6.4.

Note that this access right extension will be processed independently of whether the criticality flag is set or not. MF3E(H)x3 does not recognize any other extension. If a reader certificate with another extension marked critical is presented, it will be rejected. If the criticality flag of an unrecognized extension is not set, the extension will be ignored without rejecting the certificate.

signatureValue

The signature is a valid ECDSA signature encoded according to RFC3279 [35]. This means the signatureValue BIT STRING contains an Ecdsa-Sig-Value structure [35], which is a SEQUENCE of two INTEGERS r and s. Note that for encoding positive integers, this may lead to additional leading 0x00 byte. Therefore, the Ecdsa-Sig-Value structure can be up to 72 bytes.

7.10.2.2 Certificate Chain Validation

This section details the reader certificate chain validation as it is executed by the MF3E(H)x3 during the ECC-based mutual authentication, see Section 7.5.6. Note that the chain can also consist of a single certificate.

The inputs to this validation are:

- *targetAlgo*: ECDSA with SHA-256.
- *targetCurve*: the curve associated with the targeted KeyID.CARootKey.
- *targetPub* and *currPub*: the public key associated with the targeted KeyID.CARootKey.
- *currIssuer*: the issuer associated with the targeted KeyID.CARootKey, if any.
- *targetRestrAR*: the restricted access rights associated with the targeted KeyID.CARootKey.
- *currCert*: the current certificate to be validated, initialized to the first certificate of the received *Cert.A*.
- *currSig*: the current signature to be verified, initialized to the signatureValue of the first certificate of the received *Cert.A*.
- *currHash*: SHA-256 hash of the message to be verified, initialized to *Hash (currCert.tbsReaderCertificate)* of the first certificate of the received *Cert.A*.
- *currAR*: the current access rights, initialized to *currCert*'s ARExtensionValue, if any, else empty.

The output of a successful validation is:

- CertAccessRights: the granted access rights, as defined in Section 7.6.4.

The main steps of the certificate chain validation are as follows. On any failing check or validation step, the authentication is aborted and an error code is returned.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Salata
Quantum HiSolutions GmbH
Do not copy, share or reproduce
5fc07f79d-02c-40bc-96fe-6ed3ca82f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

1. Validate *currCert* applying the certificate validation procedure outlined below.
2. [if successful and not the last certificate in the chain]
   a. [if *currAR* is not empty] Update *targetRestrAR* with *currAR*.
   b. Update *currIssuer* with *currCert*'s subject
   c. Update *currPub* with *currCert*'s public key from subjectPublicKeyInfo
   d. Update *currCert* with the subsequent certificate received (that is, *Cert.A.leaf*).
   e. Update *currSig* with the new *currCert*'s signatureValue.
   f. Update *currHash* with *HASH* (*currCert.tbsReaderCertificate*).
   g. Start again from Step 1.
3. [if successful and last certificate in the chain]
   a. [if *currAR* is not empty] Update *targetRestrAR* with *currAR*.
   b. Update *currPub* with *currCert*'s public key from subjectPublicKeyInfo
   c. Update *currSig* with Sig.A.
   d. Update *currHash* with *HASH*(0xE0E0||OptsA||E.Pub.A||E.Pub.B).
   e. Validate *currSig* against *currHash* executing *ECDSAVerify* with *currPub*.
   f. Set CertAccessRights, that is, the granted access rights to *targetRestrAR*.

For a single or primary application, the *targetRestrAR* are the access rights associated with the primary addressed KeyID.CARootKey, that is, Bit 2-0 of P2 in ISOGeneralAuthenticate or ISOGeneralAuthenticateFinal. For a secondary application, the *targetRestrAR* are the access rights associated with the secondary addressed KeyID.CARootKey, Bit 6-4 of P2 in ISOGeneralAuthenticate or ISOGeneralAuthenticateFinal.

The certificate validation procedure consists of the following steps:

1. Parse *currCert* against the ReaderCertificate format defined above. Parsing can fail for the following reasons:
   a. *currCert* is bigger than 440 bytes.
   b. *currCert* has missing components compared to ReaderCertificatespecification. If *currCert* contains additional components, these components may either be ignored or cause a failure.
   c. *currCert*'s version is set different than 0x02, that is, no X.509 version 3 certificate.
   d. *currCert*'s serialNummer is bigger than 4-byte INTEGER.
   e. *currCert*'s signature or signatureAlgorithm is different than ECDSA with SHA-256 (OID 1.2.840.10045.4.3.2).
   f. *currCert*'s subjectPublicKeyInfo is not an ECC key (OID 1.2.840.10045.2.1) on the same curve as *targetPub*(OID 1.2.840.10045.3.1.7 or OID 1.3.36.3.3.2.8.1.1.7).
   g. *currCert* has other extensions than ARExtension that are marked critical.
   h. *currCert*'s signatureValue is not encoded according to Ecdsa-Sig-Value.
2. Validate *currSig* against *currHash* executing *ECDSAVerify* with the public key encoded in *currCert*'s subjectPublicKeyInfo.
3. [if *currIssuer* is not empty] Validate that *currIssuer* equals the *currCert*'s issuer. Note that this is more restrictive than typical X.509 certificate validation, which requires distinguishedNameMatch [8].
4. [if *currAR* is not empty] Validate *currCert*'s ARExtensionValue. Access right validation can fail for the following reasons:
   a. *currCert* holds more than one ARExtension.
   b. *currCert*'s ARExtensionValue holds unknown *ARType*.
   c. *currCert*'s ARExtensionValue format and length is not according to the *ARType* specification.
   d. *currCert*'s ARExtensionValue holds AID, which is not the currently selected single or primary application.
   e. *currCert*'s ARExtensionValue holds AID2, which is not the currently selected secondary application.
   f. *currCert*'s ARExtensionValue holds no AID2 while two applications are selected.
   g. *currCert*'s ARExtensionValue holds DFName, which is not the currently selected single application.

Provided under NDA only
COMPANY CONFIDENTIAL Personal copy for:
Original State Solutions GmbH
Do not copy or reproduce
5fc67f9d-c2fc-4e3c-90fa-6ec3ca42f1

h. *currCert*'s ARExtensionValue holds Delegated Application Management access rights while the PICC level (AID 0x000000) is not selected.
i. *currCert*'s ARExtensionValue holds access rights with a UID specified that is different from the VCUID.
j. *currCert* is a parent certificate and *currCert*'s ARExtensionValue does not have *CAdelegation* enabled.
k. *currCert* is a leaf certificate of a chain and *currCert*'s ARExtensionValue is not of the same type as *currAR*.
l. *currCert*'s ARExtensionValue KeyNoMap is not a subset of the *currAR*'s KeyNoMap.
m. *currCert*'s ARExtensionValue KeyNoMap2 is not a subset of the *currAR*'s KeyNoMap2.
n. *currCert* is the leaf certificate of a chain and *currCert*'s ARExtensionValue DAMPayLoad is not identical to the *currAR*'s DAMPayLoad.
o. *currCert* is the leaf certificate of a chain and *currCert*'s ARExtensionValueUID is not identical to the *currAR*'s UID (if any) . This means that if the parent has a UID specified, the leaf certificate must have the UID specified. If the parent has no UID specified, the leaf can still have a UID restriction.

5. [if *targetPub* has a certificate revocation file associated] Check that *currCert* is not revoked. See Certificate revocation for further details.

KeyNoMapA is a superset of *KeyNoMapB* if all bits of *KeyNoMapB* are also set in *KeyNoMapA*. In other words, *KeyNoMapB AND (KeyNoMapA XOR KeyNoMapB) == 0x0000*.

7.10.2.3 Certificate Revocation

MF3E(H)x3 supports several ways to revoke reader access in the context of ECC-based mutual and reader-unilateral authentication. First, it is possible to rotate the KeyID.CARootKey. This revokes all certificates issued by the previous KeyID.CARootKey at once. Updating of the KeyID.CARootKey is possible with ManageCARootKey, see Section 7.9.2.1.

However, the above is a drastic method, as it means that all readers must be provisioned with new certificates. As such, it does not leverage the advantages of asymmetric cryptography, and each reader (potentially) having an individual key. Therefore, MF3E(H)x3 also supports reader certificate revocation. This allows for revoking, or deny-listing of:

- individual readers, by revoking single certificates via their Certificate Serial Number (CSN).
- all certificates with a CSN lower or equal than a specified CSN value.

Certificate expiration via a certificate validity period is in general not supported by MF3E(H)x3, because the product does not maintain a notion of time.

The certificate revocation feature therefore also allows support for a form of certificate expiration by revoking all certificates with a CSN below a certain value. Under the assumption that CSNs have been assigned in sequential order, that is, with increasing expiration date, this is equivalent to expiring all certificates that have passed a certain expiration date.

This reader certificate revocation is done by maintaining a Certificate Revocation List (CRL) on the card, that is, within a FileType.StandardData or FileType.BackupData. This is similar to the CRL as defined by X.509 [8] and RFC5280 [16], but without the overhead of ASN.1 encoding and aspects like revocation dates. Using an extended variant of the WriteData, the feature also allows for flexible updating supporting Delta CRL updates (that is, only adding additional revoked certificates). CRL content can be controlled by the Issuing CA via configuration options, that is, by requiring a CRL signature and mandating monotonically increasing CRL versions (avoiding replay of old CRLs).

The effectiveness of the mechanism depends on a couple of factors depending on the use case and system integration:

- How often card interacts with (different) readers. For example, if a card only interacts with a single reader, there might be no chance to be updated with the CRL if that reader would have been compromised.

Protected under NDA only
CONFIDENTIAL - Personal copy for:
Secret Data Solutions GmbH
Quantum-safe crypto core
Do not share or reproduce
5fd07779d-8fc-085c-90fe-fec0cae421

- How fast the distribution of updated CRLs happens to the readers. For example, in installations where readers are operated offline or semioffline, this might hamper timely distribution of revocation information.
- What subset of the reader installation push updated CRLs to the card during their transaction. Note that mechanisms such as the CRL signature and CRL versioning should allow for a wide distribution across the whole reader infrastructure. Nevertheless, system integrators may opt for distribution without CRL signature by a limited set of “more trusted” readers for performance reasons.

The following sections detail how the certificate revocation:

- can be enabled, see Section 7.10.2.3.1,
- file format looks like, see Section 7.10.2.3.2,
- file can be updated, see Section 7.10.2.3.3,
- file can be read out, see Section 7.10.2.3.4.

7.10.2.3.1 Enabling Certificate Revocation

Certificate revocation needs to be enabled via the following steps:

- Via ManageCARootKey, see Section 7.9.2.1 using the parameter CRLFile and CRLFileAID. This defines the application and file number that will hold the CRLFile for this KeyID.CARootKey.
- Via CreateStdDataFile, see Section 7.12.4.1, or CreateBackupDataFile, see Section 7.12.4.2, using the parameter FileOption. When enabled, the file will have a predefined format including additional metadata, as defined in Section 7.10.2.3.2. However, the certificate revocation will only be active after detailed configuration of the feature via ChangeFileSettings. Depending on the configuration, a specified KeyID.CARootKey needs to sign any CRL update. Note that FileType.StandardData is supported at both application and PICC level. FileType.BackupData is only supported at application level.

It is possible to have the CRLFile in a different application than the KeyID.CARootKey it is associated with. This allows for use cases where multiple applications have the same KeyID.CARootKey. In this case, the KeyID.CARootKey needs to be duplicated in all applications and be associated with the correct access rights for that application, but the CRLFile can be located in one of the applications and shared with the others. Especially, when executing an authentication when multiple applications are selected, both applications need to hold the same KeyID.CARootKey, eventually also referring to the same CRLFile, see also Section 7.5.6.5.

If the file targeted by CRLFile does not exist or is not a valid CRLFile, the feature is not enabled. The behavior on authentication will be equivalent as when the feature is disabled until a valid CRLFile has been created and configured.

Once consistently enabled, reader certificates presented during an ECC-based Mutual or Reader-Unilateral authentication will be checked against the CRLFile, as defined in Section 7.10.2.3.5.

As certificate revocation could be disabled by deleting the CRLFile, one needs to be careful with granting access rights to file deletion with DeleteFile (as configured with AppKeySettings) and, in case of delegated applications, application formatting with Format. It is important that for both operations KeyID.AppMasterKey access rights (or equivalent through ECC-based authentication) are required. On top, the KeyID.AppMasterKey or certificates granting those access rights should only be used in a sufficiently secured environment (for example, from a back end or in a personalization environment).

As certificate revocation could be disabled by deleting the CRLFile, one needs to be careful with granting access rights to file deletion with DeleteFile (as configured with AppKeySettings) and, in case of delegated applications, application formatting with Format. It is important that for both operations KeyID.AppMasterKey access rights (or equivalent through ECC-based authentication) are required. On top, the KeyID.AppMasterKey or certificates granting those access rights should only be used in a sufficiently secured environment (for example, from a back-end or in a personalization environment).

Also, when sharing the CRLFile across applications, an application provider must sufficiently trust the card issuer not to delete the application holding the CRLFile. In most use cases, it is expected that sufficient trust

Provided under NXP CONFIDENTIAL - Personal copy for: Do not copy or share or transmit 5f507fed-0247-45c9-8e2c-ccfe42f1

between both parties can be expected for this not being a risk at all. However, if this trust is not there, an application deletion could be detected by the application provider if he uses the following flow: selecting two applications, and then executing an authentication over both applications and relying on some private key material within the application holding the CRLFile. The private key material can be an authentication KeyID.ECCPrivateKey, but, for example, also KeyID.ECCPrivateKey used for Transaction Signature generation. In this case, if the application holding the CRLFile was deleted and recreated with a modified or empty file, the authentication or transaction signature check would fail, as the private key would also have been deleted.

7.10.2.3.2 Certificate Revocation File Format

The CRLFile must be a FileType.StandardData or FileType.BackupData file for which the feature is enabled during file creation. When enabling a data file as a CRLFile, a predefined structure, as illustrated in Figure 15, will be applied. This structure consists of a set of metadata fields, which are not part of the actual file content, but stored as part of the file overhead. The actual file content is interpreted as a list of Certificate Serial Numbers (CSNs) with a special function for the CSN encoded at the start of the file as explained below.

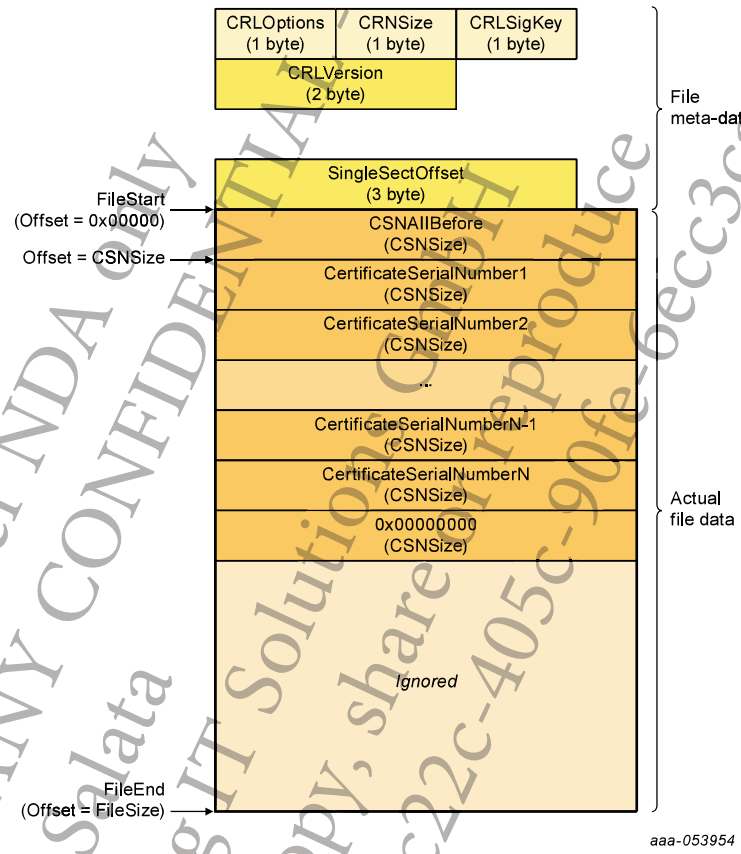


Figure 15. CRLFile format

The metadata fields can be distinguished in two classes:

- CRLOptions, CSNSize, and CRLSigKey that are configured with ChangeFileSettings, as they are expected to be configured only once during personalization. They can be retrieved with GetFileSettings.
- CRLVersion, which is configured via the WriteData, as it is expected to be changed on each CRL update. CRLVersion can be retrieved via a dedicated option on ReadData.

CRLOptions

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quartag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Only two bits of the 1-byte field CRLOptions are defined, specifying the requirements for CRLSignature and CRLVersion. The other bits are RFU and expected to be '0'.

- Bit 0: if set, both a CRLSignature and a strictly monotonically increasing CRLVersion are expected on each update of the file with WriteData. This is further defined in Section 7.10.2.3.3. Requiring a CRLSignature means that individual readers cannot control the content of the CRLFile, but that it is fully controlled by the CA. Requiring a monotonically increasing CRLVersion ensures that old CRLs cannot be replayed. If this bit is set, a CRLSigKey needs to be configured as well.
- Bit 1: if set, the updated CRLVersion must not only be strictly monotonically increasing, but exactly incremented by 1 for each update. This may be important when delta CRL updating is used, that is, to ensure that every update gets executed. If this bit is not set, any bigger CRLVersion will be accepted.
  Note: *Bit 1 can only be set if bit 0 is set.*

On file creation, by default, these bits are set to 0. This allows for initial configuration of the CRLFile during personalization without the need of a CRLSignature, for example, if this happens in a more controlled or secure environment.

As certificate revocation could be compromised by resetting these bits, one needs to be careful with granting access rights to ChangeFileSettings for the CRLFile. Preferably this operation is restricted to the keys and certificates granting FileAR.Change, which is only used from a sufficiently secured environment (for example, from a backend or in a personalization environment).

CSNSize

The size of the CSNs within the file is defined by the 1-byte field CSNSize. For MF3E(H)x3, only two values are supported:

- CSNSize= 0x00. This value is the default value and means the CRLFile is still disabled.
- CSNSize = 0x04. This value is the only value supported for actual certificate revocation, as the CSN of reader certificates is also restricted to INTEGERS of up to 4 byte, see Section 7.10.2.1.

As the CSNs from certificates are DER-encoded padding will be applied before comparison.

As certificate revocation could be disabled by setting CSNSize to 0x00, be careful when granting access rights to ChangeFileSettings for the CRLFile, see above.

CRLSigKey

If CRLOptions Bit 0 is configured, one needs to configure a CRLSigKey. The CRLSigKey needs to reference a KeyID.CARootKey that will be used for validating the CRLSignature when updating the CRLFile. This KeyID.CARootKey can be the same as the KeyID.CARootKey whose certificates can be revoked by this file, but this does not need to be the case. One can as well configure a different key, meaning that a different CA is used for the certificates and the CRLFile updating.

If CRLSigKey is not pointing to a valid KeyID.CARootKey at the time of updating the file, the update operation fails.

CRLVersion

CRLVersion is a 16-bit value encoding the current version of the CRLFile. Via the options supported by CRLOptions, this can be used to avoid replay of old CRLFile content, and also ensure that all updates are executed in case of delta CRL updating.

At file creation, the CRLVersion is initialized to 0x0000. CRLVersion is mandatory to be updated on each WriteData targeting the CRLFile.

If monotonically increasing, or even increment by one, is enforced, also updates with equal CRLVersion will not be accepted. There might be use cases where the back-end would require for proof of CRL update execution by readers in the field through the Transaction MAC or Transaction Signature feature. In general, tearing may happen between the update and the CommitTransaction required for TMAC or TSIG retrieval. In the case of

Provided under NDA only
COMPANY CONFIDENTIAL. Personal copy for:
certisalata Quantum ITS Solutions GmbH
Do not copy, share or reproduce
5f2f179d-22c-4f9f-90f6-6ec5cae42ff1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

FileType.StandardData, the update is already effective without a commit. There is however, no need to reissue the update, as reading out the (updated) CRLVersion may already be sufficient as a proof, as read commands are also included in TMAC/TSIG calculation.

Revoked CSNs

The actual file content holds the revoked Certificate Serial Numbers (CSNs). How they are interpreted depends on the position:

- Offset 0x000000: the CSN encoded at the start of the file (*CSNAllBefore* in Figure 15) revokes allCSNs with a value smaller than or equal to the encoded CSN. This allows for expiring a complete range of certificates.
- Starting from offset CSNSize: the CSNs encoded after this first position revoke individual CSNs. This part of the file will be parsed until an all zero CSN (0x00000000) is encountered. The all zero and any CSN on subsequent positions will be ignored. In the example in Figure 15 there are N revoked certificates defined, stored in the second part of the file: CertificateSerialNumber1, CertificateSerialNumber2, ... CertificateSerialNumberN (assuming none of them is set to 0x00000000).

A CA should therefore not issue a certificate with CSN 0x00000000, as this certificate cannot be revoked.

7.10.2.3.3 Certificate Revocation File Updating

Updating of a CRLFile is supported through WriteData, as defined in Section 7.13.3.2. There are a few differences, compared to the updating of regular FileType.StandardData or FileType.BackupData files:

- CRLVersion must be provided on each update. Depending on the CRLOptions configuration, the version must be strictly increasing, or even strictly increasing by one. If both Bit 0 and Bit 1 of CRLOptions are not set, any CRLVersion can be provided. If bits are set, only the CRLVersion itself is checked, MF3E(H)x3 does not do consistency checking on the content of actual revocation. This means that it is possible that a higher CRLVersion revokes fewer certificates (for example, allowing to empty the file in case of key rotation).
- a CRLSignature must be provided depending on the CRLOptions Bit 0 configuration. The signature is further defined below.

Upon successful execution of the WriteData, MF3E(H)x3 ensures that the CRLVersion and the actual file content containing the revoked CSNs are updated in one atomic operation. This means:

- for FileType.StandardData files: after successful execution of the WriteData, all values and file content are updated. In case of tearing during the WriteData, either both the metadata parameters and file content of the CRLFile have either the values from before the update or all new values.
- for FileType.BackupData files: after successful execution of the CommitTransaction, concluding the transaction as described in Section 7.14.2, all values and file content of the CRLFile are updated together with other updates from that transaction, if any. In case of tearing during the WriteData, or subsequent commands of that transaction up to CommitTransaction, no updates are committed. In case of tearing during CommitTransaction, either both the metadata parameters and the file content of the CRLFile have the values from before or all new values, similar as the other back-up files updated during the transaction.

Updating of a CRLFile with ISOUpdateBinary is not supported.

If the CRLFile is a FileType.StandardData file, there is in general risk that chained write commands will already update the file before the secure messaging MAC has been validated, see Section 7.13.3.2. This can be prevented via SetConfiguration Option 0x04. However, for CRLFile, this will be enforced by design, that is, any chained writing will always be rejected. The MF3E(H)x3 may buffer multiple chained frames and write them at once. As long as the MAC and CRLSignature (if applicable) are validated before the writing, and either all targeted data or none are updated, this is not considered as chained writing.

CRLSignature

Provided under NDA only - Personal copy for: COMPANY CONFIDENTIAL - Sebastian Gerhardt Quarg Fabian, Do not copy, share or reproduce 5fc07700-2206-4b90-6e-bec6ee42f1

The 64-byte CRLSignature is computed over the whole native WriteData. In the case of ISO/IEC 7816-4 wrapping, the wrapping parameters are ignored. The applied algorithm is ECDSA Digital Signature Generation as defined in [28]. The hash function to be applied is SHA-256, as specified in NIST FIPS 180-4 [25].

The applied key is the KeyID.CARootKey the targeted CRLFile is associated with via CRLSigKey. If the file is requiring CRLSignature, but the (referenced) KeyID.CARootKey does not exist, the WriteData shall be rejected.

The CRLSignature is calculated as follows:

$$CRLSignature = ECDSASign(K_{CARoot}, Cmd \,||\, FileNo \,||\, CRLVersion \,||\, Offset \,||\, Length \,||\, Data)$$

7.10.2.3.4 Certificate Revocation File Reading

Reading a CRLFile is supported through ReadData, as defined in Section 7.13.1.2, and ISOReadBinary, as defined in Section 7.17.3.5.

It is also possible to read out the CRLVersion with ReadData by setting Offset to 0xFFFFFF. Note that in this case Length is ignored.

In all cases, the normal access condition processing applies.

7.10.2.3.5 Certificate Revocation Checks on Authentication

Once properly enabled, see Section 7.10.2.3, certificate revocation checks are applied during ECC-based mutual and reader-unilateral authentication, see Section 7.5.6. These checks are applied on ISOGeneralAuthenticatePart2 or ISOGeneralAuthenticateFinal, depending on the flow used.

The CSN of each reader certificate (that is, *Cert.A* in case of a single certification, or *Cert.A.parent* and *Cert.A.leaf*) will be validated against the CRLFile associated with the targeted KeyID.CARootKey.

The applied checks are as follows in pseudocode:

```
for each presented reader certificate Cert do
    if Cert.CSN ≤ File[0] then
        reject Cert
    end if
    for i = CSNSize; i+ = CSNSize; i + CSNSize ≤ FileSize) do
        if (File[i] == 0x00000000) then
            stop revocation checking
        else if (File[i] == Cert.CSN ) then
            reject Cert
        end if
    end for
end for
```

The CSN at position File[i] is interpreted as being of the defined CSNSize. Also the CSN from the certificate is padded to CSNSize if needed.

If the targeted KeyID.CARootKey is associated with a CRLFile, but the file does not exist or is not a valid CRLFile, the authentication will be executed as if certificate revocation is disabled. The CRLFile may be located in a different application as configured with ManageCARootKey.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-022c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.11 Application management

MF3E(H)x3 groups user data into applications. Within an application data is further grouped into files, as described in Section 7.12.

The number of applications on a MF3E(H)x3 is only limited by the available memory. This means one can create new applications as long as there is sufficient free user memory for the application creation.

Every application is referenced by a 3-byte Application IDentifier (AID). The zero-byte AID 0x000000 is reserved for the PICC level itself, which is already present on a blank card leaving the card manufacturer.

With the ISO/IEC 7816-4 supported commands (see Section 7.17), as an alternative for the AID, an application can be referenced using a DF Name. The DF Name can be chosen independently from the AID. Both files and applications can be referenced by a 2-byte File Identifier too, using the ISOSelectFile with the option "'selection by File Identifier"'. Both options need to be enabled during application creation by providing the necessary parameters.

MF3E(H)x3 supports two types of applications:

Conventional application: This type of application is normally managed by the card issuer. The application is created with CreateApplication. Typically, this is done during card initialization and personalization. Conventional application management is detailed in Section 7.11.1.

Delegated application: This type of application can be managed by a third-party application provider. The application is created with CreateDelegatedApplication. Typically, this is done in the field without the card issuer intervening. Delegated application management (DAM) is detailed in Section 7.11.2.

The main differences between conventional and delegated application management are listed in Table 39.

Table 39. Comparison: conventional vs. delegated application management

	Conventional applications	Delegated applications
Application creation	free or requires KeyID.PICCMasterKey (depending on card configuration)	requires KeyID.PICCDAMAuthKey and cryptograms generated by card issuer with KeyID.PICCDAMENCKey and KeyID.PICCDAMMACKey
Memory consumption	No limit	Limited by quota
Memory reservation	None	Complete quota size
Memory reuse on application level	None without formatting complete PICC	Full when replaced with other delegated application of the same size
Memory reuse on file level	None for single file deletion	None for single file deletion
	None without formatting complete PICC	Full memory reuse of file space on format application

In case of conventional application creation, KeyID.PICCMasterKey access rights can be granted both through symmetric and asymmetric authentication. In case of delegated application creation, detailed delegation application access rights can be granted via the certificate, as outlined in Section 7.6.4.2.

Note that in the texts of this chapter, if it is not relevant to distinguish KeyID.NXPDAMKeys and KeyID.PICCDAMKeys, the KeyID.PICCDAMKeys are referenced.

In Section 7.11.3, it is detailed how applications can be selected. MF3E(H)x3 supports selection of up to two applications at the same time.

In Section 7.11.4, the support for information retrieval about existing applications on a card is detailed.

The required memory space depends on the selected options during creation. This is detailed in Section 7.11.5.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Salata
Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fc07790-c22c-405c-9e-ecc3cae2f1

Each application is linked to one or more sets of user-definable keys. This is specified in Section 7.8 and Section 7.9. To store user data in an application, it is necessary to create files within that application. This is further defined in Section 7.12.

7.11.1 Conventional Application Management

The conventional application management allows for application creation and application deletion. The latter can be used if an application became obsolete. The card issuer can restrict access to these operations by requiring an authentication with the KeyID.PICCMasterKey. This can be achieved by authenticating with this symmetric key via a symmetric authentication, or by an asymmetric authentication at PICC level where these KeyID.PICCMasterKey access rights are granted through the reader certificate, see Section 7.6.4.1.

7.11.1.1 CreateApplication Command

Creation of conventional applications is possible with the command CreateApplication as defined in Section 8.6.1 and Figure 48.

For application creation, the PICC level (AID 0x000000) must have been selected. Depending on the PICC Key settings, as defined in Section 7.8.4.2, an active authentication with the KeyID.PICCMasterKey may be required.

The application to be created is identified by the AID. AID 0x000000 will be rejected as it is reserved for the PICC level itself. AIDs of applications that already exist on the card are rejected.

The subsequent parameters define the key settings of the application to be created. *KeySett1* and *KeySett2* are always present.

KeySett1 defines the Application Key Settings as defined in Table 36 in Section 7.8.5.3.

KeySett2 defines additional configuration options such as the number of keys and the initial key type of the key sets of the application, see also Section 7.8.5.1.2. For ECC-only applications, *KeySett2*.Bit 3-0 can be set to 0xF to enable a temporary KeyID.AppMasterTmpKey to allow establishing initial trust with the application. For further details, see Section 7.9.5.3.

Note that in *KeySett1* or *RollKeySett* the full range of access right conditions 0x0..0xD can be used independently of the number of symmetric application keys configured with *KeySett2*. If a number is configured that is not related to a symmetric key, this means the access right can only be achieved through asymmetric authentication.

The additional optional parameter *KeySett3* is only present if Bit 4 of *KeySett2* is set.

Bit 0 of *KeySett3* will indicate presence of the following key set related parameters:

- *AKSVersion* defines the version of the initial AKS, see also Section 7.8.5.1.2.
- *NoKeySets* defines the number of key sets, which is limited to 16, see also Section 7.8.5.1.2.
- *MaxKeySize* defines the maximal key size possible in the created key sets. Note that the command is rejected if *MaxKeySize* does not allow for the key type set in *KeySett2*, see also Section 7.8.5.1.2.
- *AppKeySetSett* specifies the KeyID.AppRollKey, if any, see also Section 7.8.5.3.

Bit 2 and Bit 1 of *KeySett3* enable some additional application specific configurations. Bit 2 indicates the following can be configured at application level:

- VC configuration data with SetConfiguration Option 0x05, see Section 7.7.3.1. On creation, the PICC level values are inherited, but these can then be changed by the application later on.
- CryptoAPI configuration with SetConfiguration Option 0x15

Bit 1 indicates that KeyID.VCProximityKey can be overridden by the application. It can then be changed using ChangeKey or ChangeKeyEV2 with an active authentication of the KeyID.AppMasterKey. Note that they are

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Setup Data Management Solutions GmbH
Do not copy, share or reproduce
5fd77bd-2AC-A05c0fc6ec3cae2f1

not part of the key sets. They are initialized with the key value and version of the KeyID.PICCAppDefaultKey, as defined in Section 7.8.4.1.6.

The last two parameters *ISOFileID* and *ISODFName* are optional. These parameters are used with ISO/IEC 7816-4 file systems and the related commands (see Section 7.17).

The parameter *ISOFileID* will be used to select the application with the ISOSelectFile (selection by file identifier). Note that the values reserved by ISO/IEC 7816-4 (0x0000 0x3F00, 0x3FFF, 0xFFFF) will not be accepted.

If Bit 5 of *KeySett2* is set, every EF (elementary file), that is, files of type FileType.StandardData, FileType.BackupData, FileType.LinearRecord, FileType.CyclicRecord within the application, must be created with a 2-byte File Identifier to enable ISO/IEC 7816-4 selection with ISOSelectFile.

The parameter *ISODFName* will be used to select the application with the ISOSelectFile (selection by DF Name). The length of this parameter can be from 1 to 16 bytes. In case no DF Name is specified during creation of the application, the application cannot be referenced by a DF Name later on. Note that if a DF Name is specified, the parameter *ISOFileID* must also be present. Duplicate *ISODFName* and/or *ISOFileID*, after checking against existing applications and the PICC level VCIID, will be rejected. An *ISOFileID* that is the duplicate of a PICC-level file will also be rejected.

On application creation, the application keys of all key sets are initialized with the key value and version of the KeyID.PICCAppDefaultKey, as defined in Section 7.8.4.1.6. The key set versions of all key sets different from the initial AKS are set to 0x00.

A newly created application contains no files.

The command CreateApplication requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.11.1.2 DeleteApplication Command

Applications can be permanently deactivated with the command DeleteApplication as defined in Section 8.6.2 and Figure 49. Note that this command can be used to delete both conventional and delegated applications.

The *AID* specifies the application to be deleted. The command is rejected if the targeted AID is not found. AID 0x000000 is reserved as a reference to the PICC level itself and cannot be deleted. Therefore the command is also rejected.

For application deletion, the PICC level (AID 0x000000) or the targeted application must have been selected. It is possible that multiple applications have been selected, see Section 7.11.3, as long as the targeted application is one of them.

Depending on the PICC Key settings, see Section 7.8.4.2, a preceding authentication with KeyID.PICCMasterKey is required. If this is not the case, either an active authentication with KeyID.PICCMasterKey or an active authentication with KeyID.PICCMasterKey of the targeted application is required.

Independently of these PICC Key settings Bit 2:

- both conventional and delegated applications can be deleted with KeyID.PICCMasterKey authentication, if that was allowed during application creation via KeySett3 Bit 4 of CreateApplication, resp. CreateDelegatedApplication.
- delegated applications can always be deleted in a delegated way. In that case:
  - either, a preceding authentication with KeyID.PICCDAMAuthKey is required. Additionally, a DAMMAC is to be provided. This is further detailed in Section 7.11.2.2.2.
  - or, an asymmetric authentication is required, granting delegated application deletion via the certificate, see Section 7.6.4.2 and especially Table 22. In this case no DAMMAC needs to be presented.

Provided under NDA only
COMPANY CONFIDENTIAL
Siemens Salta
Do not copy, share or reproduce
Siemens IT Solutions GmbH
5fc5f79dc22c-40bc-90e8-6ecc3ca24f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

On successful deletion, the AID allocation is removed, therefore it is possible to create a new application with the deleted application's AID. All KeyID.AppKeys are overwritten. If the currently selected application (or one of the currently selected applications) is deleted, the PICC level (AID 0x000000) is automatically selected. As a consequence, any active authentication is aborted by moving the PICC's VCState to VCState.NotAuthenticated and any ongoing transaction is aborted.

Both conventional and delegated application can be deleted with this command. With conventional applications, the deleted memory blocks allocated for file and key storage are not released and cannot be reused. The blocks allocated for application overhead can be reused for newly created applications, see Section 7.11.5. The other blocks can only be recovered by using the command Format at PICC level, as this command releases the complete user memory space. As such, the free user memory, as reflected by FreeMem is not updated.

With delegated applications, the reserved memory blocks are released and can be reused, but only for another delegated application with exactly the same size *QuotaLimit*, targeting the same *DAMSlotNo* with a higher *DAMSlotVersion*, see Section 7.11.2.2. For this reason, on successful deletion, the *QuotaLimit*, the *DAMSlotNo* and current *DAMSlotVersion* remains associated with the reserved memory space.

The command DeleteApplication requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5. If the authentication is lost (see above), the response is sent with CommMode.Plain.

7.11.2 Delegated Application Management

7.11.2.1 Introduction

Delegated application management (DAM) allows a card issuer to delegate the application creation to third parties (application providers) in the field. Delegated applications are limited by a quota to a maximal memory consumption (*QuotaLimit*) such that the card issuer can manage the card memory space among different delegated application providers. This space is reserved on delegated application creation and cannot be allocated by other applications existing on the card. The reserved memory space is associated with a slot number and version to support the memory management by the card issuer.

After deletion, the reserved space is released and can be reused for another new delegated application of the same size targeting the same slot number with strictly greater version.

The card issuer may also decide to remove the slot version condition by using a specific pre-defined version value. Applications holding this pre-defined version number can always be (re-)installed in the targeted slot. This allows a card issuer to hand out multiple application creation rights targeting the same slot to different application providers. In this case, a card issuer cannot enforce that the memory of the targeted slot is reserved for one of the application providers. Rather the application providers must agree with the card holder in the field if they want to replace an application already occupying the slot.

This section gives a general overview of the feature. The three main steps that can be distinguished during delegated application management are described.

First, the card issuer must prepare the card for delegated application management by personalizing it with the required keys.

Then, the card issuer and an application provider must agree on the application parameters and maximal application memory consumption (that is, the *QuotaLimit*) of the new application. The card issuer will provide the application provider with the necessary keys and cryptograms (MAC, encrypted initial key) to do so.

Finally the application provider will execute the delegated application creation for which they acquired the keys and cryptograms on cards in the field. Once an application is created, they can further access and use it as a normal application as long as they remain within the assigned *QuotaLimit* for memory consumption.

These three phases are described in detail in the following subsections.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Santa Cruz Guzman
Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fc07f19-c2c4-4b3c-90fe-bec3cae4cff1

7.11.2.1.1 Personalization by card issuer

MF3E(H)x3 supports delegated application management requiring either authentication based on asymmetric cryptography, or on symmetric cryptography. The latter is compatible with the delegated application management support of MIFARE DESFire. The required preparation is for both methods is detailed in this section.

7.11.2.1.1.1 Symmetric crypto

The card issuer will generate three keys:

- KeyID.PICCDAMAuthKey: the DAM authentication key
- KeyID.PICCDAMMACKey: the DAM MAC key
- KeyID.PICCDAMENCKey: the DAM encryption Key

The KeyID.PICCDAMAuthKey will be shared with the application provider. This key can be stored UID-diversified on the card, preferably using one of the techniques in [2].

The KeyID.PICCDAMMACKey and KeyID.PICCDAMAuthKey are only known by the card issuer and stored on the PICC. These keys are not shared with the application provider. The KeyID.PICCDAMMACKey will be used to generate the MAC that will allow the application provider to create their application according to certain parameters (AID, *QuotaLimit*, and other configuration parameters).

The KeyID.PICCDAMENCKey will be used to encrypt the KeyID.AppDAMDefaultKey. This KeyID.AppDAMDefaultKey will be used to initialize the application keys of the created delegated application.

The KeyID.PICCDAMMACKey and KeyID.PICCDAMENCKey will typically not be UID-diversified, as in that case the card issuer would need to provide the application provider with a different cryptogram for each card. Therefore, also the KeyID.AppDAMDefaultKey will typically be not UID-diversified.

However note that by diversifying the KeyID.PICCDAMMACKey a card issuer can install a delegated application management scheme where they can charge the application provider on charge-per-use basis.

7.11.2.1.1.2 Asymmetric crypto

The card issuer needs to execute following steps:

- Configure the card issuer's CA root key at PICC level, to allow to authenticate the application provider via ECC-based authentication.
- Ensure availability of (Priv.PICC, Pub.PICC) and related certificate to allow the application provider to authenticate the MF3E(H)x3 via ECC-based authentication.

In the default configuration, no CA root key is set at PICC level. The card issuer can configure its public key as detailed in Section 7.9.2.

For card authentication, the Cert.PICC can be used to authenticate the MF3E(H)x3. This only allows the application provider to identify the card as a valid MF3E(H)x3. Therefore, it is recommended that the card issuer provisions its own certificate. This can be done by either overwriting the Cert.PICC, or by adding an additional certificate file at PICC level, see Section 7.10.1.2. If the card issuer, provides the application provider with the related CA public key, it allows the application provider to identify the MF3E(H)x3 as being a valid card issued by the card issuer.

7.11.2.1.2 Application creation approval between card issuer and application provider

When the card issuer sells memory space to an application provider they must agree on all the application parameters, including the AID and the maximal consumable memory size (*QuotaLimit*), as defined for CreateDelegatedApplication, see Section 7.11.2.2.

Provided under NXP CONFIDENTIAL - Personal copy for: Sergio Data Quantum IT Solutions GmbH. Do not copy, share or reproduce 5fc07719d-c2c-405c-90fe-6ecc-ca4211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

If the application of a delegated application provider became obsolete, it is possible to reuse the memory space of this application for a new application of the same size (QuotaLimit). Therefore the application parameters of a delegated application will include a number (DAMSlotNo), allowing a card issuer to target a specific slot of reserved delegated application memory space, and a version information (DAMSlotVersion), allowing to define the chronological order of the applications that can occupy a slot. This also prevents a delegated application to be re-created once it became obsolete.

7.11.2.1.2.1 Symmetric crypto

The card issuer will generate a MAC (that is, the *DAMMAC*) over the agreed application parameters using the KeyID.PICCDAMMACKey.

The card issuer supplies the application provider with the following:

- KeyID.PICCDAMAuthKey. If this key is UID-diversified on the PICC, the master key is supplied to the application provider such that they can generate themselves the UID-diversified derivatives for communication with the individual cards.
- Application parameters together with the *DAMMAC* over these parameters generated by the KeyID.PICCDAMMACKey.
- In case of Random ID and diversified KeyID.PICCDAMAuthKey, a non-diversified application key is supplied. This allows the application provider to retrieve the UID.
- KeyID.AppDAMDefaultKey, which will be used to initialize the application keys of the delegated application, and this key encrypted with the KeyID.PICCDAMENCKey. Note that the KeyID.AppDAMDefaultKey could also be chosen by the application provider and provided to the card issuer for encryption.

7.11.2.1.2.2 Asymmetric crypto

The card issuer will provision the application provider with a reader certificate, granting delegated application management access rights, see Section 7.6.4. For a certificate allowing delegated application creation, this includes the agreed application parameters.

Depending on the application provider's needs, a leaf certificate can be provided, or the card issuer can allow further CA delegation. In the latter, a parent certificate is provided and the application provider can act as an intermediate CA to generate reader certificates.

Depending on the certificate access rights, the card issuer can allow delegated application management for his full card population, i.e. all cards that hold the public key of the key pair used to sign the certificates as CA root key. Alternatively, it can just grant access rights for a single card, by including the VCUID, allowing charge-per-use schemes.

Note that in the asymmetric setting, the application provider can independently chose the KeyID.AppDAMDefaultKey, at the time of the actual application creation. Therefore, it does not need to be known by the card issuer.

7.11.2.1.3 Application creation by application provider

When the application provider wants to create the application on the PICC, they take the following steps.

For a delegated application the PICC needs to store the maximal consumable memory size (QuotaLimit). This maximal memory size will be checked by the PICC on each file creation.

7.11.2.1.3.1 Symmetric crypto

- (if needed) Retrieve the UID with the non-diversified application key.
- Authenticate with the KeyID.PICCDAMAuthKey. A UID-diversified derivative is used if the key has been diversified on the PICC by the card issuer.

Provided under NDA ONLY - Personal copy for: COMPANY CONFIDENTIAL Sergii Salata, Quantum IT Solutions GmbH Do not copy, share or reproduce 5fc07790-c22c-4055-99fe-6ecc5ae421

- Create the delegated application providing the agreed parameters and the received *DAMMAC*
- Personalize the newly created delegated application. This should include replacing all keys initialized with the KeyID.AppDAMDefaultKey with new key values only known to the application provider and preferably UID-diversified.

7.11.2.1.3.2 Asymmetric crypto

- Execute an ECC-based mutual authentication presenting the recieved reader certificate (or a chain complemented with an own generated leaf certificate).
- Create the delegated application providing the agreed parameters, providing an KeyID.AppDAMDefaultKey only known to the application provider. This may be a temporary key if targeting an ECC-only application, see Section 7.9.5.3.
- Personalize the newly created delegated application.

7.11.2.1.4 Application creation through MIFARE 2GO

To ease delegated application management, NXP developed a webservice called MIFARE 2GO. The MIFARE 2GO platform allows card issuers and application providers to connect to each other, and make the application provider's applications available for the card issuer's cards. Applications can then be loaded onto card's by the card holder through the platform using a mobile app.

To further smoothen the process, an option is foreseen where the MIFARE 2GO platform can make use of a set of pre-configured KeyID.NXPDAMAuthKey. This avoids that the card issuer must pre-personalize the card with their own KeyID.PICCDAMKeys, and share those with the MIFARE 2GO platform, or even share their KeyID.PICCMasterKey with MIFARE 2GO to allow the platform to configure the KeyID.PICCDAMKeys. Instead the KeyID.NXPDAMAuthKey are already trust-provisioned to the card during the manufacturing by NXP. They just must be enabled by the card issuer by configuring PICCKeySettings Bit 4.

Note that in the rest of this chapter, if it is not relevant to distinguish KeyID.NXPDAMAuthKey and KeyID.PICCDAMKeys, the KeyID.PICCDAMKeys are referenced.

7.11.2.1.5 Application deletion

7.11.2.1.5.1 Symmetric crypto

To also allow delegated application deletion, an option is foreseen to also issue DeleteApplication after authentication with the KeyID.PICCDAMAuthKey, additionally protected by a *DAMMAC*.

The main use case of this is when the delegated application management is done through MIFARE 2GO, as it allows removing of applications triggered by the card holder through the platform.

Note that in use cases where the KeyID.PICCDAMAuthKey is handed out to multiple not fully trusted and/or competing parties, this delegated application deletion might create a denial-of-service risk.

7.11.2.1.5.2 Asymmetric crypto

Also in the asymmetric setting, application deletion can be delegated, see Section 7.6.4.

7.11.2.2 CreateDelegatedApplication Command

Creation of delegated applications is possible with the command CreateDelegatedApplication. It consists of two parts, as defined in Section 8.6.3 and Figure 50. Note that most error checking is done after having received the second part of the command (as defined in the tables).

Provided under NDA/MTA - Personal copy for: COMPANY NAME OF NDA/MTA Signatory Solutions GmbH Do not copy, share or reproduce 5fc07bd-c22-40-c-90fe-6ec3-ca92f1

For delegated application creation, the PICC level (AID 0x000000) must have been selected. An active authentication with the KeyID.PICCDAMAuthKey is required.

This command is rejected if one of the required PICC level keys (KeyID.PICCDAMAuthKey, KeyID.PICCDAMMACKey and KeyID.PICCDAMENCKey) are still disabled, see Section 7.8.4.1.2.

In case of delegated application management with asymmetric crypto, an active asymmetric authentication at PICC level is required, granting delegated application creation access rights via the certificate.

The application to be created is identified by the AID. AID 0x000000 will be rejected as it is reserved for the PICC level itself. AIDs of applications that already exist on the card are rejected, except if it is the AID of the application currently occupying the targeted *DAMSlotNo*. In other words, it is possible to replace an application by a new one holding the same AID (similar for *ISOFileID* and *ISODFName*).

The parameters *DAMSlotNo* and *DAMSlotVersion* indicate the slot number and version associated with the memory space of the delegated application. This allows further management of this memory space by the card issuer.

The parameter *QuotaLimit* indicates the maximal memory consumption of the application to be created in number of 32 byte blocks. The command is rejected if *QuotaLimit* is not sufficient for the overhead for application administration and key storage, which depends on the application parameters and key settings passed. This overhead is reduced from the available requested QuotaLimit and the remaining blocks are available for file creation.

The other application parameters *KeySett1*, *KeySett2*, the optional *KeySett3* and Key Set parameters, and the ISO/IEC 7816-4 related parameters (*ISOFileID* and *ISODFName*) are interpreted as with CreateApplication (see Section 7.11.1.1).

The command is sent in two frames, using the DESFire native chaining. The command is split after the optional *ISODFName* parameter. *EncK* and *DAMMAC* are sent in the second frame CreateDelegatedApplicationPart2.

The *DAMMAC* of CreateDelegatedApplicationPart2 is the MAC as received from the card issuer to allow application creation. It was created with the KeyID.PICCDAMMACKey, according to the method defined in Section 7.11.2.2.2.

The application will only be created after successful validation of the received *DAMMAC* by the PICC using the KeyID.PICCDAMMACKey. If this MAC validation fails, the command is rejected.

The parameter *EncK* contains the encrypted KeyID.AppDAMDefaultKey as defined in Section 7.11.2.2.1.

The delegated application will be created only after successful validation of *DAMMAC* and decryption of *EncK*. In VCState.AuthenticatedECC, that is, if preceded by an asymmetric authentication, only the parameter InitK is provided. Similar as *EncK* this consists of the KeyID.AppDAMDefaultKey, but in this case it is encrypted by the CommMode.Full secure messaging, that is, with KeyID.SesAuthENCKey.

The delegated application will be created as follows:

- if the targeted *DAMSlotNo* is already associated with a memory space on the card (i.e. there had already been a delegated application created before targeting the same *DAMSlotNo*), this memory space is reused for the new application. The command is rejected if
  - the targeted *DAMSlotNo* is associated with a different *QuotaLimit* size
  - the given *DAMSlotV* version is different from 0xFF and the given *DAMSlotV* version is not strictly greater than the *DAMSlotVersion* currently associated with the memory space.

  If the application is successfully created, an active application currently occupying the targeted DAMSlotNo is automatically replaced. This means the behavior is exactly as if this application is deleted before creating the new application. Note that in case of tearing, the card may reach a state where the old application is deleted with DeleteApplication without the new one being created, see also Section 7.7.
- if the targeted *DAMSlotNo* does not exist, by reserving free memory space of the requested *QuotaLimit* size. The command is rejected if there is insufficient free memory

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

On successful creation, the targeted *DAMSlotNo* is associated with the given *DAMSlotVersion*. This way the *DAMSlotVersion* automatically prevents re-creation of obsolete delegated applications with lower *DAMSlotVersion* targeting the same *DAMSlotNo*. Note also that re-creation of the current delegated application (that is, with equal *DAMSlotVersion*) is not possible. This allows the card issuer to delete delegated applications using DeleteApplication without necessarily immediately reusing the slot for a new delegated application. The application owner will not be able to re-install his deleted application anymore. The only exception is if the *DAMSlotVersion* is set to 0xFF. Any application with *DAMSlotVersion* set to 0xFF, can always be created, replacing any existing application inside the targeted slot, as long as the new application is fulfilling all other conditions for delegated application creation.

There are no restrictions on the order of creation of applications coming from *DAMSlotNo*. This means applications targeting different *DAMSlotNo* can be created in any order.

On successful creation, the memory space of requested *QuotaLimit* size is reserved uniquely for this delegated application. During the whole lifetime of this application, the reserved memory space cannot be used by any other application.

On application creation, the application key values and versions are initialized with the KeyID.AppDAMDefaultKey. This includes the application specific KeyID.VCProximityKey if enabled at application level (see *KeySett3*). Note that 16-byte keys (KeyType.AES128) are initialized with the first 16 bytes of $K_{AppDAMDefault}$.

A newly created application contains no files. The command CreateDelegatedApplication requires CommMode.Full. Note that in case of symmetric authentication, this is still backwards compatible with MIFARE DESFire, where the command was defined as requiring CommMode.MAC, as only the new parameter InitK which is only present after asymmetric authentication will be encrypted by the secure messaging.

Note that the command is handled as a chained command, so the secure messaging MAC will be sent with CreateDelegatedApplicationPart2. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.11.2.2.1 AppDAMDefaultKey and its encryption

The following cases can be distinguished:

- If the AKS targets KeyType.AES128 (*KeySett2:Bit 7-6* is set to '01'): the KeyID.AppDAMDefaultKey consists of a 24-byte string $K_{AppDAMDefault}$ encoding the key (independent of the targeted key type) and a related 1-byte key version KeyVerAppDAMDefault. This is for backwards compatibility reasons with MIFARE DESFire. Note that only the first 16 byte of $K_{AppDAMDefault}$ are used to initialize the KeyType.AES128 keys.
- If the AKS targets KeyType.AES256 (*KeySett2: Bit 7-6* is set to '11'): the KeyID.AppDAMDefaultKey consists of a 32-byte string $K_{AppDAMDefault}$ encoding the key (independent of the targeted key type) and a related 1-byte key version $KeyVer_{AppDAMDefault}$.

7.11.2.2.1.1 Symmetric crypto

The KeyID.AppDAMDefaultKey is encrypted by the card issuer using the KeyID.PICCDAMENCKey and provided to the application provider. The application provider will pass it as the *EncK* parameter of CreateDelegatedApplicationPart2 to the PICC.

Note that the encryption of the KeyID.AppDAMDefaultKey is independent of the active secure messaging mode.

For encryption, $K_{AppDAMDefault}$ and $KeyVer_{AppDAMDefault}$ are padded by adding random bytes in front of the key.

The resulting string is encrypted with the KeyID.PICCDAMENCKey to form the parameter *EncK* of CreateDelegatedApplicationPart2. Encryption and decryption are calculated using the underlying block cipher according to the CBC mode of NIST SP800-38A [17]. The initialization vector is the zero byte string.

Provided under NDA to CONFIDENTIAL - Personal copy for: Digital Data Solutions GmbH. Document copy, share or reproduce 5fc0791-d2c-4fc-90fe-6ccccae2ff1

The notation $E_{DAM}(key,message)$ is used to denote the encryption and $D_{DAM}(key,message)$ for decryption. The parameter *EncK* of CreateDelegatedApplicationPart2 is then defined as follows:

$$EncK = E_{DAM}(K_{PICCDAMENC}, Random(7) \,||\, K_{AppDAMDefault}(24) \,||\, KeyVer_{AppDAMDefault}(1))$$

Only the first 16 byte of $K_{AppDAMDefault}$ are used. Note that still 24 bytes are provided for backwards compatibility reasons with MIFARE DESFire.

Targeting KeyType.AES256, the parameter *EncK* of CreateDelegatedApplicationPart2 is 48 byte and defined as follows:

$$EncK = E_{DAM}(K_{PICCDAMENC}, Random(15) \,||\, K_{AppDAMDefault}(32) \,||\, KeyVer_{AppDAMDefault}(1))$$

7.11.2.2.1.2 Asymmetric crypto

The KeyID.AppDAMDefaultKey is independently chosen by the application provider and passed via the parameter *InitK*. This parameter is a CmdData parameter and therefore encrypted by the secure messaging session.

Targeting KeyType.AES128, the plain *InitK*. is defined as follows:

$$InitK = K_{AppDAMDefault}(24) \,||\, KeyVer_{AppDAMDefault}(1))$$

Only the first 16 byte of $K_{AppDAMDefault}$ are used.

Targeting KeyType.AES256, the plain *InitK* is defined as follows:

$$InitK = K_{AppDAMDefault}(32) \,||\, KeyVer_{AppDAMDefault}(1))$$

7.11.2.2.2 DAMMAC Calculation

7.11.2.2.2.1 Symmetric crypto

The *DAMMAC* is the cryptogram prepared by the card issuer and provided to the application provider for execution of CreateDelegatedApplicationPart2.

Note that the *DAMMAC* calculation is independent of the active secure messaging mode.

The used key is the KeyID.PICCDAMMACKey defined at PICC level. The MAC calculation algorithm depends on the key type of the targeted KeyID.PICCDAMMACKey:

- KeyID.PICCDAMMACKey of KeyType.AES128 or KeyType.AES256: see EV2 Secure Messaging Section 7.5.10.3 with zero byte IV

A different notation than the one for Secure Messaging MACs is used: $MACt_{DAM}(key,message)$ is used to denote the CMAC operation (including truncation if applicable).

The *DAMMAC* is calculated as follows with all the preceding fields in the command as input:

$$DAMMAC = MACt_{DAM}(K_{PICCDAMMAC}, Cmd \,||\, AID \,||\, DAMSlotNo \,||\, DAMSlotVersion \,||\, QuotaLimit \,||\, KeySett1 \,||\, KeySett2[\,||\, AKSVersion \,||\, NoKeySets \,||\, MaxKeySize \,||\, RollKey][\,||\, ISOFileID][\,||\, ISODFName] \,||\, EncK)$$

with *Cmd* being the command code of CreateDelegatedApplication, that is, the first part.

7.11.2.2.2.2 Asymmetric crypto

In case of asymmetric authentication, *DAMMAC* is not sent, as the application parameters are already authenticated via the reader certificate access rights.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Gala
Quantas IT Solutions GmbH
Do not copy, share or reproduce
5fc079dc-2c-405c-90fe-6ecc3cae4211

7.11.2.3 Delegated application deletion

As already mentioned in Section 7.11.1.2, delegated applications can also be deleted in a delegated way with DeleteApplication.

In case of asymmetric crypto, an asymmetric authentication is required, granting delegated application deletion via the certificate, see Section 7.6.4.2 and especially Table 22. In this case no *DAMMAC* needs to be presented. This section further details the symmetric crypto case.

An active authentication with the KeyID.NXPDAMAuthKey is required. Alternatively, depending on PICCKeySettings Bit 4, authentication with KeyID.NXPDAMAuthKey is required. In this case, the related KeyID.NXPDAMMACKey is used as well. This is not further mentioned explicitly in this section. This option is always possible for delegated applications, independently of PICCKeySettings Bit 2.

After KeyID.PICCDAMAuthKey authentication, the command is rejected if the targeted application is not a delegated application, that is, it was not created with CreateDelegatedApplication.

The *DAMMAC* of DeleteApplication is the MAC created with the KeyID.PICCDAMMACKey, according to the method defined in Section 7.11.2.2.2.

The application will only be created after successful validation of the received *DAMMAC* by the PICC using the KeyID.PICCDAMMACKey. If this MAC validation fails, the command is rejected.

For the rest, the processing is as described in Section 7.11.1.2.

7.11.2.3.1 DAMMAC Calculation

Note that the *DAMMAC* calculation is independent of the active secure messaging mode.

The used key is the KeyID.PICCDAMMACKey defined at PICC level. The MAC calculation algorithm depends on the key type of the targeted KeyID.PICCDAMMACKey:

- KeyID.PICCDAMMACKey of KeyType.AES128 or KeyType.AES256: see EV2 Secure Messaging Section 7.5.8.2 with zero byte IV

A different notation than the one for Secure Messaging MACs is used: $MACt_{DAM}(key,message)$ is used to denote the CMAC operation (including truncation if applicable).

The *DAMMAC* is calculated as follows with all the preceding fields in the command as input:

$$DAMMAC = MACt_{DAM}(K_{PICCDAMMAC}, Cmd \,||\, AID)$$

with *Cmd* being the command code of DeleteApplication.

7.11.2.4 Quota Limit Handling

On the PICC, the memory consumption of delegated applications is limited by the *QuotaLimit* set during application creation. During the whole application lifetime, the PICC tracks the memory consumption of the application.

At application creation with CreateDelegatedApplication, the PICC already adds up the application overhead for application configuration and key storage. Note that this overhead is fixed at application creation.

After application creation, the memory consumption quota is managed as follows:

- at file creation, see Section 7.12.4, the memory consumption of the delegated application is increased with the memory allocation of the targeted file type. If the quota limit is exceeded, the file creation fails.
- at file deletion, see Section 7.12.5, the memory consumption of the delegated application does not change, as the allocated memory blocks are not released.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

- at application formatting, see Section 7.7.4.2, the memory consumption of the delegated application is decreased with the total number of memory blocks that had been allocated to files. The memory consumption is the same as just after application creation.

7.11.3 Application selection with SelectApplication command

Applications can be selected with SelectApplication prior to execute any operations on them. Applications are identified with their AID specified in Section 7.11. One or two applications can be selected at the same time. Refer to Section 7.11.3.1 for two applications selection specification.

Alternatively applications can be selected with ISOSelectFile, see Section 7.17.3.4.

Once an application has been selected, all subsequent operations are applied on it. If the operation is not supported by the application, then the card responds with a PERMISSION_DENIED.

When an application is selected, the current active authentication status is aborted and the card is put in VCState.NotAuthenticated state. Therefore depending on the operations to be executed on the application a new authentication might be needed.

Selecting an application automatically abort any open transaction and start a new one. Refer to Section 7.14 for transaction specification.

When trying to select an application and the TMCLimit of the targeted application was reached, see Section 7.14.3.2.2, the application is still selected, but OPERATION_OK_LIM is returned to indicate limited functionality: CommitTransaction is rejected, that is, no further transactions can be executed.

Also, when the application holds a FileType.TransactionMAC configured for Transaction Signature, but the KeyID.AppTransactionSIGKey, that is, the referenced KeyID.ECCPrivateKey, does not exist or is not enabled for Transaction Signature, OPERATION_OK_LIM is returned and the same limited functionality applies.

7.11.3.1 Shared Application Management

The notion of shared application is introduced in order to allow a transaction to group operations on multiple files from two different applications.

In a SelectApplication a second application can be specified as a parameter through its AID. One application, coming at first place in the command, is called the primary application and the other, coming at second place in the command, is called the secondary one. The operations on files from both selected applications take then part of the same transaction.

Once two applications have been selected, operation on keys or files could take place in the primary or secondary application. The Second Application Indicator (SAI) is used to tag the key numbers or file numbers in order to distinguish in which application the operation should take place. The SAI is defined as 0x80 and is xored to the key numbers or file numbers.

When two applications are selected, data management operations on both applications are executed within the same transaction. Using asymmetric authentication, see Section 7.5.6, it is possible to gain access rights for both application at the same time. The presented reader certificate can encode this as explained in Section 7.6.4.1. During the authentication a referenced CA Root Key, see Section 7.9.2, which is shared by both applications, must be addressed. Using the symmetric authentication, see Section 7.5.8, the applications are independent. In that case, operations performed on an application follow the same rules as if the application would have been selected alone. For example it is not possible to use an authentication with the secondary application KeyID.AppMasterKey to perform operations requiring an authentication with the KeyID.AppMasterKey on the primary application. Only one of the two selected applications can have a FileType.TransactionMAC file (see Section 7.14.3).

For example if an authentication with key number 4 of the second application is needed to access a file with file number of 0x0A the following flow of command is needed:

Provided under NDA/NTAL - Personal copy for: CONFIDENTIAL Quantum Solutions GmbH. Original data source: share.nxp.com. Do not copy or reproduce. 5fc07f99d-5f2c-4b5c-a01e-6ec3cae2211

- Authentication with key number 0x84
- ReadData with file number 0x8A

Note that with an ECC-based authentication it is possible to gain access rights over both applications by a single authentication. For this the addressed KeyID.CARootKey must be the same for both applications. During the authentication then a certificate granting access rights to the two applications must be presented, see Section 7.6.4.1 and especially Table 20. For example, if ACMap2 includes access condition at bit 4, this means equivalent access rights as key number 4 in the secondary application, and therefore the file number 0x0A can be accessed with a ReadData targeting 0x8A, similarly as above.

7.11.3.2 SelectApplication Command

SelectApplication as defined in Section 8.6.4 and Figure 51, selects one or two applications through their AIDs. It aborts any current open transaction and starts a new one. It also aborts any active valid authentication and puts the card in VCState.NotAuthenticated state.

If SelectApplication is rejected and returns an error, then the PICC level is selected by default. The command is rejected if:

- one or both applications are not present in the card.
- two applications are selected and one of them is the PICC level (AID1 = 0x000000 OR AID2 = 0x000000).
- two applications are selected and both have the same AID (AID1 = AID2).
- two applications are selected and both application holds a FileType.TransactionMAC file. Refer to Section 7.12.1.5.
- One of the selected applications holds a FileType.TransactionMAC file which TMC has already reached the maximal value 0xFFFFFFFF. Refer to Section 7.12.1.5.

Note that ISO/IEC ISO7816-4 standard commands will always target the primary application, as defined in Section 7.17.3. ISO/IEC ISO7816-4 wrapped commands, see Section 7.17.2, can of course target both using the same approach as native commands.

7.11.4 Application Information Retrieval

7.11.4.1 GetApplicationIDs Command

The AID list of all active applications can be retrieved with the command GetApplicationIDs as defined in Section 8.6.5 and Figure 52.

This command does not accept any parameters.

If the PICC level (AID 0x000000) is not selected, the command is rejected.

If Bit 1 of PICCKeySettings is not set, see Table 33, one of the following authentications is required:

- a symmetric authentication with the KeyID.PICCMasterKey or the KeyID.PICCDAMAuthKey (resp. KeyID.NXPDAMAuthKey)
- an asymmetric authentication granting KeyID.PICCMasterKey access rights
- an asymmetric authentiation granting delegated application management access rights

As response the PICC sends a sequence of all installed AIDs. If this sequence does not fit into one single frame, native command chaining is applied, see Section 7.5.2.2. Note that a single AID is never split across two frames. If there are not installed active AIDs, OPERATION_OK is returned without any data.

Depending on PICCKeySettings, this command can be issued without an active authentication, but if there is an active authentication the command GetApplicationIDs requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc077bd-022c-405c-90fe-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Provided under NDA by Setup Data GmbH
COMPANY CONFIDENTIAL Personal copy for:
Quantas Technologies GmbH
Do not share or reproduce
5fc07790-c22c-405c-96de-6ecc3cae42a1

7.11.4.2 GetDFNames Command

The AID, *ISOFileID* and *ISODFName* of all active applications with ISO names can be retrieved with the command GetDFNames as defined in Section 8.6.6 and Figure 53.

This command does not accept any parameters.

If the PICC level (AID 0x000000) is not selected, the command is rejected.

If Bit 1 of PICCKeySettings is not set, see Table 33, one of the following authentications is required:

- a symmetric authentication with the KeyID.PICCMasterKey or the KeyID.PICCDAMAuthKey (resp. KeyID.NXPDAMAuthKey)
- an asymmetric authentication granting KeyID.PICCMasterKey access rights
- an asymmetric authentiation granting delegated application management access rights

As response the PICC sends the AID, *ISOFileID* and *ISODFName* of any active application having the ISO/IEC 7816-4 identifiers as set during creation, see Section 7.11.1.1 or Section 7.11.2.2. If *ISOFileID* and *ISODFName* were not set on creation, the application is not returned on this command. For each of these applications, the data is sent in a single frame using the native chaining, see Section 7.5.2.2.

Depending on PICCKeySettings, this command can be issued without an active authentication, but if there is an active authentication the command GetDFNames requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.11.4.3 GetDelegatedInfo Command

To support a card issuer with the delegated application management, GetDelegatedInfo allows getting specific information associated with delegated applications.

The specific parameters associated with a delegated application slot, can be retrieved with the command GetDelegatedInfo as defined in Section 8.6.7 and Figure 54.

The *DAMSlotNo* is taken as an input parameter. The command is rejected if the targeted *DAMSlotNo* is not found on the card. This means no delegated application targeting this *DAMSlotNo* has been created on the card.

The PICC level (AID 0x000000) application must have been selected.

If Bit 1 of PICCKeySettings is not set, see Table 33, one of the following authentications is required:

- a symmetric authentication with the KeyID.PICCMasterKey or the KeyID.PICCDAMAuthKey (resp. KeyID.NXPDAMAuthKey)
- an asymmetric authentication granting KeyID.PICCMasterKey access rights
- an asymmetric authentiation granting delegated application management access rights

The command GetDelegatedInfo requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

The returned parameters *DAMSlotVersion*, *QuotaLimit*, *Freeblocks* and *AID* have the values associated with the targeted delegated application slot. The returned AID is set to 0x000000 if the targeted delegated application slot exists on the card, but is not occupied (as its application has been deleted with DeleteApplication). In this case, *DAMSlotVersion* and *QuotaLimit* are still set to the actual values related with the targeted slot. *FreeBlocks* is then set to the same value as *QuotaLimit*, as the complete slot is free.

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.11.5 Memory Consumption

Memory allocation is done in 32-byte blocks, see Section 7.7.4.

Memory consumption for applications is

- General: 1 block
- Additional for delegated applications: 1 block
- Additional: 1 block, if at least one of the following conditions is fulfilled:
  - created application is created with ISO/IEC 7816-4 parameters ISOFileID or ISODFName (similar for delegated applications)
  - create application holds application-specific VC capability data or keys, that is, Bit 1 or Bit 2 of *Keysett3* is set.
- Additional: 1 block, if enabling the AuthCtr with SetConfiguration Option 0x16

Therefore, at most 4 blocks are consumed. These overhead blocks may be partially or completely reused after application deletion with DeleteApplication.

Note that also the memory required for key storage is already allocated on application creation (including VC keys and/or configuration data), as defined in Section 7.8.7. This memory is not reused after application deletion with DeleteApplication.

Memory consumption of file create is defined in Section 7.12.6.

From the overall PICC perspective, delegated applications consume immediately the full *QuotaLimit* on CreateDelegatedApplication as this memory is reserved and allocated for the created application. From application perspective, 2 blocks are already consumed as application overhead. The remainder of *QuotaLimit* is available for key storage (also allocated on application creation), see Section 7.8.7, and file creation, see Section 7.12.6.

Provided under NDA ONLY - Personal copy for:
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6e22cae021

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.12 File management

MF3E(H)x3 maintains user data into files of specific types listed in Section 7.12.1. Files are managed through creation, deletion, and information retrieval and update functions respectively specified in Section 7.12.4, Section 7.12.5, Section 7.12.3, and Section 7.12.2.3. File access can be restricted with an access right management specified in Section 7.12.2.

7.12.1 File Types

MF3E(H)x3 supports the following types of data storage:

- raw data as specified in Section 7.12.1.1 and Section 7.12.1.2
- a value, i.e. a 4 byte signed integer as specified in FileType.Value
- record data as specified in Section 7.12.1.4.2 and Section 7.12.1.4.3
- transaction MAC or Signature related data as specified in Section 7.12.1.3 and Section 7.12.1.5

All file type but FileType.StandardData implements a backup mechanism that is specified in the following sections.

All MF3E(H)x3 files are defined with a file number and the communication mode that has to be used when accessing the file data. The file number is coded over 1 byte. It is unique per file in an application. The communication mode is defined in Section 7.5.4.

7.12.1.1 FileType.StandardData

FileType.StandardData stores the data as raw data byte per byte. Data is accessed by chunk of byte at a certain offset in the data file and with a certain length in byte.

A FileType.StandardData file is created with CreateStdDataFile, see Section 7.12.4.1.

A FileType.StandardData file can be read with ReadData and ISOReadBinary. The data can be written with WriteData and ISOUpdateBinary.

FileType.StandardData is defined by its size in bytes.

FileType.StandardData is not covered by the transaction mechanism and therefore does not implement any backup mechanism. Therefore data are available to read as soon as they are written in the file. Limited anti-tearing protection is foreseen, as at least single frames of a WriteData and ISOUpdateBinary are ensured to be written anti-tearing protected, see Section 7.13.1.2 and Section 7.17.3.6 for more details.

A FileType.StandardData file can potentially hold a certificate, as defined in Section 7.10.1.1, or a certificate revocation list, as defined in Section 7.10.2.3.2.

FileType.StandardData is the only file type that can exist at the PICC level (AID 0x000000). In the default configuration, there are two pre-created files holding NXP trustprovisioned certificates, as defined in Section 7.10.1.2 and Section 7.10.1.4.

7.12.1.2 FileType.BackupData

FileType.BackupData provides the same feature as a FileType.StandardData file but it adds a backup mechanism to it.

A FileType.BackupData file is created with CreateBackupDataFile, see Section 7.12.4.2.

All operations executed on a FileType.BackupData within the current transaction will be effective only when a successful CommitTransaction will be executed. Meanwhile or if the transaction is aborted or not successfully committed, the previous file image remains effective and is returned by a reading of the file.

Provided under NDA only - Personal copy for: COMPANY/CONFIDENTIAL - Seegi Santa Clarita Solutions GmbH Quanting TI Solutions GmbH Do not copy, share or reproduce 5fc07f79dc22c4039ef0fe6ecc3ca42f

The backup mechanism implemented on FileType.BackupData implies that such files consume twice memory with regard to memory used for data storing than FileType.StandardData.

Similarly as FileType.StandardData, a FileType.BackupData file can potentially hold a certificate, as defined in Section 7.10.1.1, or a certificate revocation list, as defined in Section 7.10.2.3.2.

7.12.1.3 FileType.Value

FileType.Value stores the data as 4 bytes signed integer.

A FileType.Value file is created with CreateValueFile, see Section 7.12.4.3.

A FileType.Value file can be read with GetValue. The value can be manipulated with Credit, Debit and LimitedCredit respectively.

FileType.Value is restricted by an upper and lower limit that the current value of the file cannot exceed.

FileType.Value can support a limited credit feature as defined in Section 7.13.2.4.

The free get value, or *GetFreeValue*, allows a user to get freely the value of a FileType.Value bypassing the access rights authentication if required as long as there is one access condition associated with GetValue that is different from no access condition.

FileType.Value implements a backup mechanism. All operations executed on a FileType.Value within the current transaction will be effective only when a successful CommitTransaction will be executed. Meanwhile or if the transaction is aborted or not successfully committed, the previous file image remains effective and is returned by a GetValue.

7.12.1.4 Linear and Cyclic Record Files

FileType.LinearRecord and FileType.CyclicRecord group raw data into records. FileType.LinearRecord and FileType.CyclicRecord files are defined by the size of records and the maximum number of records a file can contain.

A FileType.LinearRecord file is created with CreateLinearRecordFile, see Section 7.12.4.4.

A FileType.CyclicRecord file is created with CreateCyclicRecordFile, see Section 7.12.4.5.

A record file can be read with ReadRecords and ISOReadRecord. Records can be written and updated with WriteRecord and ISOAppendRecord, and with UpdateRecord respectively. The file can be cleared again with ClearRecordFile.

7.12.1.4.1 Record Indexing

Within a record file, records are numbered starting with the latest record written. Meaning that the latest record written has the number 0, the second latest record written has the number 1 and so on. Its valid range is from 0 to the number of existing records *ExistingRecCount - 1* in the record file.

This indexing is used in the data management commands to address records within records file.

Note however that when reading out records, the records are returned in the chronological order that they have been written, that is, the oldest record is returned first.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Sala
Quantum Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22-4056-9fe-6ecc3ae2f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.12.1.4.2 FileType.LinearRecord

A FileType.LinearRecord file adds up new records when a successful WriteRecord is executed until the maximum number of records of the file is reached. Then it is not possible anymore to write new records into the file.

This means one additional record is required to temporarily store the updated record before the CommitTransaction is executed. Therefore, one additional record should be created on top of the desire number of records if the UpdateRecord command is to be used. This additional record should not be written and the record file should never be full. Once the FileType.LinearRecord file is full, that is, the maximum record has been written, it is not possible to update any records anymore.

FileType.LinearRecord implements a backup mechanism. All operations executed on a FileType.LinearRecord within the current transaction will be effective only when a successful CommitTransaction is executed. Meanwhile or if the transaction is aborted or not successfully committed, the previous file image remains effective and is returned by a ReadRecords.

7.12.1.4.3 FileType.CyclicRecord

A FileType.CyclicRecord file adds up new records when a successful WriteRecord is executed up when the maximum number of records of the file is reached. Then the next new record will overwrite the oldest record as defined in Section 7.12.1.4.1.

FileType.CyclicRecord implements a backup mechanism. All operations executed on a FileType.CyclicRecord within the current transaction will be effective only when a successful CommitTransaction will be executed.

If the transaction is aborted or not successfully committed the previous file image is effective and returned by a ReadRecords.

Note that the maximum number of records must be one larger than the application requires, as the backup mechanism in case of cycling consumes one record.

7.12.1.5 FileType.TransactionMAC

A FileType.TransactionMAC file is used to store Transaction MAC-related information, see Section 7.14.3. It consists of the following items:

Transaction MAC Counter (TMC) for Transaction MAC and Signature. 4 byte unsigned integer, as specified in Section 7.14.3.2.1. The TMC is initialized to 0x00000000 on file creation. This value means that no Transaction MAC/Signature has been computed yet. Note that this value will never be used in a Transaction MAC/Signature computation.

Transaction MAC Value (TMV) for Transaction MAC. 8 bytes, as specified in Section 7.14.3.2.4. The TMV is initialized to all zero bytes on file creation.

Transaction Signature Value (TSV) for Transaction Signature. 64 bytes, as specified in Section 7.14.4.2.4. The TSV is initialized to all zero bytes on file creation.

A FileType.TransactionMAC file is created with CreateTransactionMACFile, see Section 7.12.4.6. An application can hold only one FileType.TransactionMAC file.

A FileType.TransactionMAC file can be read with ReadData, protected by the FileAR.Read access right, see below. The behavior is identical as reading a FileType.BackupData file of 12 bytes for Transaction MAC, and 68 bytes for Transaction Signature.

A FileType.TransactionMAC file cannot be written directly. The TMC and TMV/TSV are related with the latest successful transaction and are updated automatically on successful CommitTransaction.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

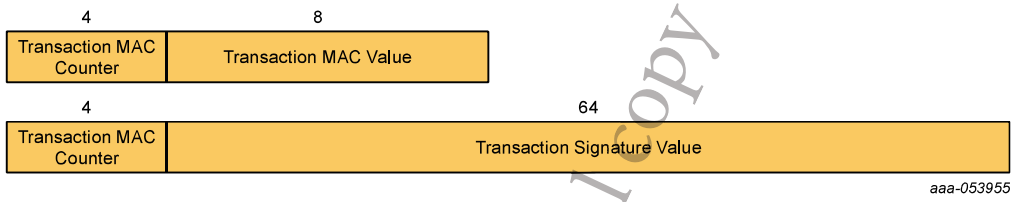


Figure 16. Transaction MAC file content

The file layout is shown in Figure 16. The TMC and TMV/TSV are related with the latest successful transaction and are updated on CommitTransaction. Therefore, the implementation of this file type needs to support the integrated backup mechanisms required for anti-tearing protected atomic update of these values together with the rest of the transaction updates.

Note that compared to other file types, when deleting and re-creating a FileType.TransactionMAC file, the memory occupied by this file will be reused, if possible. No additional memory will be allocated if the newly created file fits in the already allocated space of the deleted file, see also Section 7.12.6.

7.12.1.5.1 Access Rights

The access rights management of a FileType.TransactionMAC file differs from the general access right management outlined in Section 7.12.2 for other file types.

A FileType.TransactionMAC cannot have additional access rights, but will only get a mandatory set of access rights, see Section 7.12.2. These mandatory access rights have the meaning defined in Table 16. Note that FileAR.Write is RFU and expected to be set to 0xF.

Table 40. TransactionMAC access rights

Access right	Bit Index	Value	
FileAR.Read	15-12	condition as in Table 16	
FileAR.Write	11-8	RFU (that is, 0xF)	
FileAR.ReadWrite	7-4	KeyID.AppCommitReaderIDKey	
		0x0..0xD	CommitReaderID enabled, requiring authentication with the specified key index or granting equivalent access rights via asymmetric authentication
		0xE	CommitReaderID enabled with free access
		0xF	CommitReaderID disabled.
FileAR.Change	3-0	condition as in Table 16	

For details on the CommitReaderID behavior, as specified with FileAR.ReadWrite, see Section 7.14.3.4.4.

7.12.2 File Access Rights Management

For a generic introduction on access right management, see Section 7.6, especially Section 7.6.1, for the encoding of access conditions.

File data is accessed with 3 different access rights FileAR.Read, FileAR.Write and FileAR.ReadWrite. Each of these access rights is permitting the use of a subset of commands defined in Section 7.12.2.2.

In addition, an access right called FileAR.Change is specified per file permitting ChangeFileSettings to change the file access rights.

Provided under NDA only. COMPANY CONFIDENTIAL - Personal copy for: Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

An access right is granted if at least one condition associated to it is satisfied. Such conditions are called access conditions.

Each file is associated with 1 up to 8 access conditions sets. Each set is containing three access conditions: one for each of FileAR.Read, FileAR.Write and FileAR.ReadWrite. The first set is called the mandatory set as it is always present.

FileAR.Change is associated with 1 access condition per file.

This is represented in Figure 17.

CreateTransactionMACFile always restrict its number of access condition sets to 1 set and has a different semantics for the mandatory access rights. Refer to Section 7.12.1.5.1.

At file creation a mandatory set of access conditions associated with FileAR.Read, FileAR.Write and FileAR.ReadWrite access rights plus a condition associated with the FileAR.Change access right are specified. These access conditions can be changed using ChangeFileSettings. A parameter specifying whether the file restricts the number of access conditions sets to 1 is specified too but cannot be changed thereafter.

When a file does not restrict its number of access conditions sets to 1, additional sets of access conditions can be specified with ChangeFileSettings.

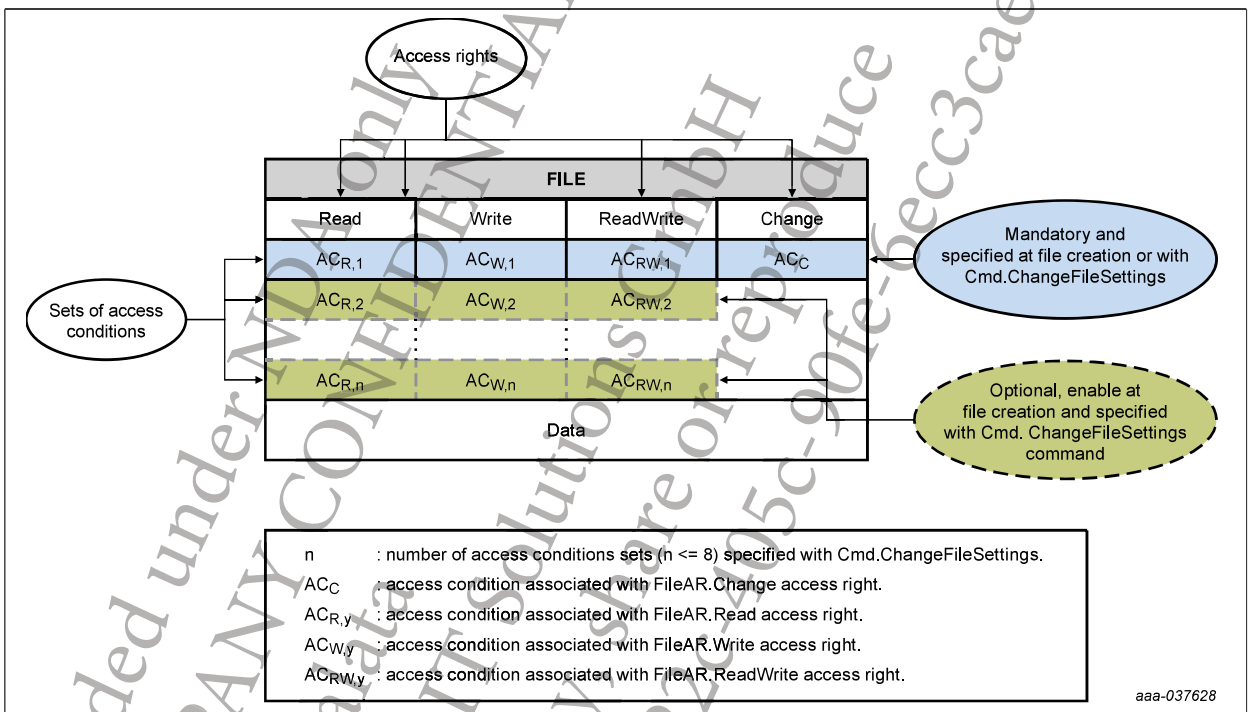


Figure 17. MIFARE DESFire file access rights associated with access conditions grouped in sets

If a file creator is only interested in using multiple access conditions for one particular access right and not the others, he can set the access conditions in which he has no interest to no access.

The set of access conditions are coded on 2 bytes as shown in Table 41. RFU access conditions are expected to be set to 0xF (for future extensibility)

Table 41. Set of Access condition coded on 2 bytes

Bit Index	Description	Value
15..12	FileAR.Read	access condition as in Table 16

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

Table 41. Set of Access condition coded on 2 bytes...*continued*

Bit Index	Description	Value
11..8	FileAR.Write	access condition as in Table 16.
7..4	FileAR.ReadWrite	access condition as in Table 16.
3..0	FileAR.Change or RFU	access condition as in Table 16. FileAR.Change for the 1st mandatory set of access condition else RFU (that is, 0xF).

Note: *For PICC-level files, see Section 7.10.1.2, access conditions are interpreted as in Table 17.*

7.12.2.1 Secure Dynamic Messaging related access rights

Additionally, a FileType.StandardData file can be associated with the following Secure Dynamic Messaging access rights: FileAR.SDMMetaRead, FileAR.SDMFileRead and FileAR.SDMCtrRet.

This is only possible if SDM was enabled on file creation, see CreateStdDataFile.

The FileAR.SDMCtrRet is interpreted as the access rights defined above, according to Table 16 and grants access to GetFileCounters. The others have a different interpretation.

The FileAR.SDMMetaRead is a bit special as it does not define access to certain commands, that is, by setting this access right one does not affect the policy on when certain commands will be allowed or not. It purely defines the mirroring of PICCData, that is, whether the PICCData will be mirrored in plain, encrypted or not at all, see also Section 7.5.11.1.3. This is interpreted according to Table 42.

Table 42. FileAR.SDMMetaRead values

Condition value	Description
0x0..0xD	KeyID.SDMMetaReadKey: key number of an KeyID.AppKey used to encrypt the PICCData before mirroring
0xE	Plain PICCData mirroring
0xF	No PICCData mirroring

The FileAR.SDMFileRead and FileAR.SDMFileRead2 will, as soon as one of them is different from 0xF, grant free access to ReadData and ISOReadBinary.

The FileAR.SDMFileRead, as defined in Table 43, allows configuring a symmetric KeyID.AppKey. This key is used to derive session keys, see Section 7.5.11.1.12. KeyID.SesSDMFileReadMACKey is used for SDMMAC computation as defined in SDMMAC and Section 7.5.11.1.9.KeyID.SesSDMFileReadENCKey is used for file data encryption, see SDMENCFileData as defined in Section 7.5.11.1.6 and Section 7.5.11.1.7.

The FileAR.SDMFileRead2, as defined in Table 43, allows configuring an asymmetric KeyID.ECCPrivateKey. This key is used for SDMSIG computation as defined in SDMSIG and Section 7.5.11.1.11. If both FileAR.SDMFileRead and FileAR.SDMFileRead2 configure a key, a SDMSIG is computed with the key of FileAR.SDMFileRead2. No SDMMAC is calculated in this case, but the FileAR.SDMFileRead will still be used for encryption if enabled. The Table 45 gives an overview of the possible combinations.

Table 43. FileAR.SDMFileRead values

Condition value	Description
0x0..0xD	KeyID.SDMFileReadKey: free access, key number of an KeyID.AppKey that is to be applied for the Secure Dynamic Messaging
0xE	RFU

Protected under NDA of NXP Semiconductors GmbH
COMPANY CONFIDENTIAL, presented to:
Sergiu Galatanu
Quantum Solutions
Do not copy, share or reproduce
5fc0779d-622-905c-90fe6eccsae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 43. FileAR.SDMFileRead values...*continued*

Condition value	Description
0xF	No Secure Dynamic Messaging for Reading

Table 44. FileAR.SDMFileRead2 values

Condition value	Description
0x0..0xD	KeyID.SDMFileRead2Key: free access, key number of an KeyID.ECCPrivateKey that is to be applied for the SDMSIG calculation
0xE	RFU
0xF	No Secure Dynamic Messaging for Reading

Table 45. FileAR.SDMFileRead and FileAR.SDMFileRead2 combinations

FileAR.SDMFileRead	FileAR.SDMFileRead2	SDMENCFileData	SDMMAC	SDMSIG	Comment
KeyID.AppKey	valid KeyID.ECCPrivateKey	Yes, mandatory to be enabled	No	Yes	-
KeyID.AppKey	invalid KeyID.ECCPrivateKey	Yes, mandatory to be enabled	No	No	Rejected at ChangeFile Settings. If KeyID.ECCPrivateKey gets invalidated afterwards, the static file data is returned at SDMMACOffset.
KeyID.AppKey	0xF	Yes, if enabled	Yes	No	-
0xF	valid KeyID.ECCPrivateKey	No	No	Yes	-
0xF	invalid KeyID.ECCPrivateKey	No	No	No	Rejected at ChangeFile Settings. If KeyID.ECCPrivateKey gets invalidated afterwards, the static file data is returned at SDMMACOffset.
0xF	0xF	No	No	No	-

Note that granting FileAR.SDMFileRead and FileAR.SDMFileRead2 is independently from whether or not free access is granted via FileAR.Read or FileAR.ReadWrite. If FileAR.SDMFileRead or FileAR.SDMFileRead2 is granted, the Secure Dynamic Messaging will always be applied in VCState.NotAuthenticated.

If FileAR.SDMFileRead and FileAR.SDMFileRead2 is put to 0xF, it is still possible to freely read the file if FileAR.Read or FileAR.ReadWrite are put to 0xE. In this case, plain mirroring of the PICCData, see Section 7.5.11.1.3, is still applied if the card is configured for that.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07790-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.12.2.2 Access right association with commands

In Table 46, it is listed to which commands the access rights are granting access to.

Table 46. Command list associated with access rights

FileAR.Read	FileAR.Write	FileAR.ReadWrite	FileAR.Change	FileAR.SDMMetaRead	FileAR.SDMFileRead or FileAR.SDMFile Read2	FileAR.SDMCtrRet
ReadData	WriteData	ReadData	ChangeFileSettings	-	ReadData	GetFileCounters
GetValue	GetValue	WriteData	VDE_WriteData		ISOReadBinary	
Debit	Debit	GetValue				
ReadRecords	LimitedCredit	Debit				
ISOReadBinary	WriteRecord	Credit				
ISOReadRecord	ISOAppendRecord	LimitedCredit				
VDE_ReadData	ISOUpdateBinary	ReadRecords				
	VDE_WriteData	WriteRecord				
		UpdateRecord				
		ClearRecordFile				
		ISOReadRecord				
		ISOAppendRecord				
		ISOReadBinary				
		ISOUpdateBinary				
		CommitReaderID				
		VDE_ReadData				
		VDE_WriteData				

A command listed in Table 46 is accepted if at least one access condition associated with an access right (could be several) granting access to it is satisfied. If authenticated, and the only access conditions satisfied are free access 0xE within FileAR.Read, FileAR.Write, FileAR.ReadWrite and FileAR.Change, then the CommMode.Plain is to be applied.

If not authenticated, Secure Dynamic Messaging will be applied if access is granted via FileAR.SDMFileRead or FileAR.SDMFileRead2, even if there is free access via one of the other access rights. FileAR.SDMFileRead and FileAR.SDMFileRead2 are not affecting the regular secure messaging, that is, if authenticated.

A command listed in Table 46 is rejected if there is no satisfied access conditions associated with an access right (could be several) granting access to it. The command returns:

- PERMISSION_DENIED if all access conditions associated with all access rights granting access to the command are denying access.
- AUTHENTICATION_ERROR if at least one access condition associated with one of the access rights granting access to the command requires a valid authentication, while being in VCState.NotAuthenticated, or in VCState.AuthenticatedAES but authenticated with the wrong key.
- CERT_ERROR if at least one access condition associated with one of the access rights granting access to the command requires a valid authentication, while being in VCState.AuthenticatedECC but not having obtained the required access rights from the targeted KeyID.CARootKey or reader certificate presented during the authentication.

Personal copy for: Sergey Sitnikov, Quantum Solutions GmbH
Do not copy, share or reproduce
5fc0779d-2f2c-4059-9016-6ecc3cae42f1
PRELIMINARY CONFIDENTIAL

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.12.2.3 ChangeFileSettings Command

ChangeFileSettings, as defined in Section 8.7.12 and Figure 66, permits updating the communication mode of a file as specified in Table 7 and the access rights of a file by means of all its sets of access conditions as specified in Table 41.

The *AccessRights* parameter is mandatory and updates the mandatory set of access conditions as shown in Figure 17 and defined in Table 41.

The parameters *NrAddARs* and *AddAccessRights* are optional and update the second up to *NrAddARs* optional sets of access conditions as shown in Figure 17 and defined in Table 41. Note that previous additional access conditions are always cleared, even if *NrAddARs* and *AddAccessRights* are not present. For files at the PICC level (AID 0x000000), additional access rights are not supported.

ChangeFileSettings also allows enabling the Secure Dynamic Messaging and mirroring features, see Section 7.5.11 for more details. However, note that this is only possible on FileType.StandardData files, where SDM was enabled on creation. For files at the PICC level (AID 0x000000), SDM is not supported.

If targeting a FileType.TransactionMAC file, when changing FileAR.ReadWrite, that is, enabling or disabling CommitReaderID, this will not affect the currently ongoing transaction.

If targeting a FileType.TransactionMAC file, ChangeFileSettings supports setting the TMCLimit, see Section 7.14.3.2.2. When enabling TMCLimit, the currently ongoing transaction is aborted. Individual files can be excluded from Transaction MAC or Transaction Signature calculation by setting FileOption Bit 6 and defining the related TMIExclFileMap. This is further detailed in Section 7.14.3.4.3 and Section 7.14.4.4.3.

If targeting a CRLFile, CRLOptions and CSNSize as defined in Section 7.10.2.3.2 must be provided.

The command is rejected if:

- adding sets of access conditions when a file has been created restricting its sets of access conditions to one.
- adding sets of access conditions to a PICC-level file.
- the *FileNo* parameter does not refer to an existing file in the selected application.
- the SAI is set and no second application is selected.
- the FileAR.Change is not granted because it is a no access 0xF
- the FileAR.Change is not granted because it requires an authentication with a KeyID.AppKey which is currently not active, or an asymmetric authentication where the certificate grants the access rights of the encoded key number.
- trying to enable Secure Dynamic Messaging on a file where it is not supported (for example, as not pre-enabled during creation).
- trying to enable Secure Dynamic Messaging for a PICC-level file.
- the provided configuration for Secure Dynamic Messaging and mirroring is inconsistent according to the conditions of Section 7.5.11, as reflected in Section 8.7.12..
- Trying to set a TMCLimit which is smaller or equal to the current TMC.
- Trying to enable CommitReaderID while there was no KeyID.AppTransactionMACKey configured on creation of the FileType.TransactionMAC.

Under active authentication, the command ChangeFileSettings requires CommMode.Full. There is one exception: if FileAR.Change of the targeted file is configured to 0xE allowing free access, also under active authentication CommMode.Plain is to be applied. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

Provided under NDA only. Personal copy for: CONFIDENTIAL Sergio Salazar Quantag Technologies GmbH. Do not copy, share or reproduce 5fc07790-c22c-405c-90fe-6ecc3e4211

7.12.3 File Information Retrieval

7.12.3.1 GetFileSettings Command

GetFileSettings, as defined in Section 8.7.10 and Figure 64, allows getting information on the properties of a specific file. The information provided by this command depends on the type of the file which is queried.

The file from which the settings have to be retrieved is defined by *FileNo* specified over 5 bits. *FileNo* can also indicate if a file in the secondary application is targeted via the SAI bit.

The first part of the returned message is the same for all file types:

- the actual file type, see Section 7.12.1.
- the communication mode as specified in Table 7.
- the access rights of a file by means of all its sets of access conditions as specified in Table 41.

All subsequent bytes in the response have a special meaning depending on the file type:

- FileType.StandardData and FileType.BackupData: file size over 3 bytes. If Secure Dynamic Messaging applies for the targeted FileType.StandardData file, this is also indicated, and the related offset parameters are returned. If targeting a CRLFile, CRLOptions and CSNSize are returned. Note that in the latter case, the file cannot be configured for SDM.
- FileType.Value: three fields, each of four bytes length, are returned whereby the first field returns the lower limit of the file (as defined at file creation), the second field returns the upper limit (as defined at file creation) and the next field returns the current maximum limited credit value according to the latest successfully committed transaction, see Section 7.13.2.4. So if this value is already affected by the currently ongoing transaction, this is not reflected in the response of GetFileSettings. If the limited credit functionality is not in use, the last field contains all zeros. The last byte codes whether LimitedCredit and free GetValue is enabled for this file (as defined at file creation).
- FileType.LinearRecord and FileType.CyclicRecord files: three fields, each of three bytes length, are returned whereby the first field codes the size of one single record (as defined at file creation), the second field codes the maximum number of records within the record file (as defined at file creation) and the last field returns the current number of records within the record file. This number equals the maximum number of records which currently can be read, see Section 7.12.1.4.1.
- FileType.TransactionMAC file: KeyID.AppTransactionMACKey related info like the current version is returned. If enabled, also the current TMCLimit is returned. If not returned, this means the feature is disabled and TMCLimit is set to the maximum value. If file exclusion is enabled, it is also returned which files are excluded. If configured for Transaction Signature, the key number of the KeyID.AppTransactionSIGKey is also returned.

The command is rejected if:

- the SAI is set and no second application is selected.
- the targeted file does not exist
- depending on AppKeySettings, see Table 36, an authentication is reuqired. This can either be a symmetric authentication with the KeyID.AppMasterKey, or an asymmetric authentication where the CertAccessRights grant the equivalent access rights, that is, Bit 0 of the KeyNoMap is set.

Under active authentication, the command GetFileSettings requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.12.3.2 GetFileCounters Command

GetFileCounters, as defined in Section 8.7.11 and Figure 65, supports retrieving of the following counter values:

- current values associated with the 24-bit SDMReadCtr related with a FileType.StandardData file after enabling Secure Dynamic Messaging, see Section 7.5.11 and Section 7.12.2.3.

Provided under NXP COMPANY CONFIDENTIAL - Personal copy for: Segi Santa Quantum Solutions GmbH Do not copy, share or reproduce 5fc779d-cc2c-4052-90ff-6e6e3ae2ef1

The command is rejected if

- The PICC level (AID 0x000000) is selected
- the targeted file does not exist
- the targeted file is not a FileType.StandardData file
- if targeting a FileType.StandardData file, depending on FileAR.SDMCtrRet, permission is always denied or requires authentication.

Under active authentication, the command GetFileCounters requires CommMode.Full for SDMReadCtr retrieval. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.12.3.3 GetFileIDs Command

GetFileIDs, as defined in Section 8.7.8 and Figure 62, returns the complete list of file IDs of all active files of the selected application.

The command takes no parameters.

If two applications are selected, only the file identifiers for the primary application are returned.

Each File ID is coded in one byte and is in the range from 0x00 to 0x1F. Duplicate values are not possible as each file must have an unambiguous identifier. As the number of files is limited to 32 within one application, the response always fits into one single data frame.

The command is rejected if:

- depending on AppKeySetSettings, see Table 36, an authentication is required. This can either be a symmetric authentication with the KeyID.AppMasterKey, or an an asymmetric authentication where the CertAccessRights grant the equivalent access rights, that is, Bit 0 of the KeyNoMap is set.

Under active authentication, the command GetFileIDs requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.12.3.4 GetISOFileIDs Command

GetISOFileIDs, as defined in Section 8.7.9 and Figure 63, returns the complete list of the 2 byte ISO/IEC 7816-4 File IDentifiers of all active files within the currently selected application.

The command takes no parameters.

If two applications are selected, only the file identifiers for the primary application are returned.

Each File ID is coded in two bytes. Duplicate values are not possible as each file must have an unambiguous identifier.

As the number of files is limited to 32 within one application, the response not always fits into one single data frame.

If the number of bytes to send to the PCD does not fit into one single frame, the PICC waits for a status frame with status byte 0xAF, before the next frame is sent to the PCD.

The command is rejected if:

- depending on AppKeySettings, see Table 36, an authentication is required. This can either be a symmetric authentication with the KeyID.AppMasterKey, or an an asymmetric authentication where the CertAccessRights grant the equivalent access rights, that is, Bit 0 of the KeyNoMap is set.
- the application was created with *ISOFileIDs* disabled

The command GetISOFileIDs requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

Provided under NDA only - Personal copy for: COMPANY CONFIDENTIAL
Gerhard Santa, Quantas IT Solutions GmbH
Do not copy, share or reproduce
5f0f7bd-222c-495c-90fe-62c3cfe42f1

7.12.4 File Creation

Some properties of file creation commands hold for all of them. They are listed first, while file type-specific aspects are listed in the related subsections.

The file creation commands all share the following parameters: *FileNo*, *FileOption* and *AccessRights*.

The *FileNo* encodes the file number in the range of 0x00 to 0x1F which the new created file should get within the currently selected application. If the file number is already occupied, the file creation fails.

The *FileNo* can indicate the shared application is targeted for file creation via the Shared Application Indicator, see Section 7.11.3.1. If the SAI is set and no second application is selected, the command is rejected.

If Bit 5 of *KeySett2* was set during application creation (CreateApplication or CreateDelegatedApplication), every EF (elementary file), that is, files of type FileType.StandardData, FileType.BackupData, FileType.LinearRecord, FileType.CyclicRecord within the application, must be created with a 2-byte File Identifier *ISOFileID* to enable ISO/IEC 7816-4 selection with ISOSelectFile. Note that the values reserved by ISO/IEC 7816-4 (0x0000 0x3F00, 0x3FFF, 0xFFFF) will not be accepted.

The *FileOption* defines whether additional access rights can be used with the newly created file, see Section 7.12.2. Note that for FileType.TransactionMAC it is not possible to allow additional access rights.

Next to this, *FileOption* also defines the communication mode of the targeted file, see Section 7.5.4.

The *AccessRights* define the mandatory access right set of the newly created file. Note that the meaning of these access rights depends on the targeted file type, see Section 7.12.2.2. For FileType.TransactionMAC, the interpretation is different, as defined in Section 7.12.1.5.1.

MF3E(H)x3 supports file creation for FileType.StandardData files not only within an application, but also at the PICC level (AID 0x0000000). These FileType.StandardData files require an ISOFileID. Duplicaes, also with existing applications (DF), are rejected. Additional access rights are not supported for files at the PICC level. Depending on the AppKeySettings, see Section 7.8.5.3: an authentication is required. This can either a symmetric authentication with the KeyID.AppMasterKey, or an asymmetric authentication where the CertAccessRights grant the equivalent access rights, i.e. Bit 0 of the KeyNoMap is set.

If a conventional application is targeted, that is, an application that was created with CreateApplication, a file creation command is rejected if not enough free user memory is available for the memory consumption of the new file.

If a delegated application is targeted, that is, an application that was created with CreateDelegatedApplication, a file creation command is rejected if the memory consumption of the new file would make the application exceed its *QuotaLimit*.

Under active authentication, all file creation commands require CommMode.MAC, except for CreateTransactionMACFile which requires CommMode.Full. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.12.4.1 CreateStdDataFile Command

General aspects of file creation, shared by all file creation commands, are described at the start of Section 7.12.4.

MF3E(H)x3 supports file creation for FileType.StandardData files not only within an application, but also at the PICC level (AID 0x0000000).

In addition to the parameters listed above, CreateStdDataFile, as defined in Section 8.7.1 and Figure 55, specifies the size of the file in bytes. *FileSize* is defined as an 3 byte integer. The file will be initialized with all zero bytes. If the file is intended for Secure Dynamic Messaging, see Section 7.5.11, then this needs to be pre-enabled at file creation via FileOption Bit 6. However, note that this is not supported for file 0x1F, as this one is

Provided under NDA - PERSONAL COPY for: CONFIDENTIAL - Personal copy for: Sergii Salamatin, Quantas IT Solutions GmbH, Do not copy, share or reproduce 5fc07790-c22c-a05c-90fe-ec3cae02f1

typically used as FCI, see Section 7.17.3.4. At PICC level, Secure Dynamic Messaging is not supported. If the file is intended to be used as a CRLFile, this must be pre-enabled at file creation via FileOption Bit 4.

7.12.4.2 CreateBackupDataFile Command

General aspects of file creation, shared by all file creation commands, are described at the start of Section 7.12.4.

In addition to the parameters listed above, CreateBackupDataFile, as defined in Section 8.7.2 and Figure 56, specifies the size of the file in bytes. *FileSize* is defined as an 3 byte integer. The file will be initialized with all zero bytes.

If the file is intended to be used as a CRLFile, this must be pre-enabled at file creation via FileOption Bit 4.

7.12.4.3 CreateValueFile Command

General aspects of file creation, shared by all file creation commands, are described at the start of Section 7.12.4.

In addition to the parameters listed above, CreateValueFile, as defined in Section 8.7.3 and Figure 57, specifies the lower and upper value limit, the initial value, and the limited credit and *GetFreeValue* enable options of the file. *LowerLimit*, *UpperLimit*, and *Value* are defined as 4 byte signed integers. *LimitedCreditEnabled* is defined over 1 byte with the following definition:

- Bit 0: Enable LimitedCredit feature, see LimitedCredit in Section 7.13.2.4. Here 0 means that LimitedCredit is disabled and 1 enables this feature.
- Bit 1: Enable GetFreeValue feature, which allows free access for reading the value file as specified in FileType.Value. Here 0 means that Free GetValue is disabled, and 1 enables this feature.

The upper limit has to be higher than or equal to the lower limit, otherwise an error message would be sent by the PICC and therefore the file would not be created. The initial value of the value file is also checked by the PICC, in case of inconsistency, the file is not created, and an error message is sent by the PICC.

7.12.4.4 CreateLinearRecordFile Command

General aspects of file creation, shared by all file creation commands, are described at the start of Section 7.12.4.

In addition to the parameters listed above, CreateLinearRecordFile, as defined in Section 8.7.4 and Figure 58, specifies the record size and the maximum number of records of the file. *RecordSize* and *MaxNoOfRecs* are defined as 3 bytes unsigned integer. Both have to be in the range from 0x000001 to 0xFFFFFF. After creation, the file will hold no records.

7.12.4.5 CreateCyclicRecordFile Command

General aspects of file creation, shared by all file creation commands, are described at the start of Section 7.12.4.

In addition to the parameters listed above, CreateCyclicRecordFile, as defined in Section 8.7.5 and Figure 59, specifies the record size and the maximum number of records of the file.

RecordSize and *MaxNoOfRecs* are defined as 3 bytes unsigned integer. *RecordSize* has to be in the range from 0x000001 to 0xFFFFFF, while *MaxNoOfRecs* has to be in the range from 0x000002 to 0xFFFFFF. Note that the maximum number of records needs to be one larger than the application requires, as the backup mechanism in case of cycling consumes one record. After creation, the file will hold no records.

Provided under NDA only. COMPANY CONFIDENTIAL - Personal copy for: Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-405c-90fe-6ecc3cae42f1

7.12.4.6 CreateTransactionMACFile Command

Creation of a FileType.TransactionMAC file is possible with the command CreateTransactionMACFile as defined in Section 8.7.6 and Figure 60.

General aspects of file creation, shared by all file creation commands, are described at the start of Section 7.12.4.

An application can hold only one FileType.TransactionMAC file. If multiple applications are selected only one of them can hold a FileType.TransactionMAC file. So file creation will fail if one of the currently selected applications already holds of file of this type. If the FileType.TransactionMAC file is re-created in an application that already had a deleted FileType.TransactionMAC file before (with no formatting in between), no new user memory is allocated, if the targeted configuration requires equal or less memory than the previous file, see also Section 7.12.6.

In case of Transaction MAC, or Transaction Signature with authenticated CommitReaderID (i.e. FileAR.ReadWrite is different from 0xE or 0xF), *TMKeyOption*, *TMKey*, and *TMKeyVer* define the KeyID.AppTransactionMACKey. The key type is defined by *TMKeyOption*. For the KeyID.AppTransactionMACKey, MF3E(H)x3 only supports KeyType.AES128 and KeyType.AES.256. The key value is set to the value specified in *TMKey* the length of this parameter depends on the targeted key type. *TMKeyVer* specifies the key version.

In case of Transaction Signature, TSIGKeyNo specified the KeyID.ECCPrivateKey within the targeted application which is to be used as KeyID.AppTransactionSIGKey. Note that if this key does not exist or is not enabled for Transaction Signature, an error is returned.

On the file creation, the initial TMRIPrev encoding the ReaderID returned with CommitReaderID, is set to all zero bytes, see also Section 7.14.3.2.5.

Creating the FileType.TransactionMAC file, automatically aborts an ongoing transaction. A new transaction is started, see Section 7.14.2.2.

Individual files can be excluded from Transaction MAC or Transaction Signature calculation by setting *FileOption* Bit 6 and defining the related *TMIExclFileMap*. This is further detailed in Section 7.14.3.4.3 and Section 7.14.4.4.3.

The TMCLimit is by default disabled after CreateTransactionMACFile. This is equivalent to putting it to the maximum value: 0xFFFFFFFF.

Note that by creating this file, the Transaction MAC feature is enabled for the targeted application as specified in Section 7.14.3.3. The KeyID.AppTransactionMACKey can be updated by deleting and re-creating the file.

7.12.5 File deletion with DeleteFile command

Deletion of a file is possible with the command DeleteFile as defined in Section 8.7.7 and Figure 61. The operation of this command invalidates the file directory entry of the specified file which means that the file cannot be accessed anymore.

The file to be deleted is defined by *FileNo* specified over 5 bits. If the SAI is set and no second application is selected, the command is rejected.

At application level, depending on the AppKeySettings, see Section 7.8.5.3, a preceding authentication is required. This can either be a symmetric authentication with the KeyID.AppMasterKey, or an asymmetric authentication where the CertAccessRights grant the equivalent access rights, i.e. Bit 0 of the KeyNoMap is set. At PICC level, authentication with the KeyID.PICCMasterKey, or an an asymmetric authentication where the CertAccessRights grant the equivalent access rights, is always required.

Independently whether a conventional or a delegated application is targeted, a file deletion command does not affect the memory consumption of the targeted application. This means the allocated memory excluding the

Provided under NDA to COMPANY CONFIDENTIAL Quantag IT Solutions GmbH Daniel Santa Cruz Anton Personal copy for: Do not copy, share or reproduce 5f0f7f91-022-4055-9016-6ec3ae2f1

file overhead, see Section 7.12.6, cannot be immediately reused. The file overhead can be reused for newly created files within the same application. The exception is FileType.TransactionMAC, where both overhead and file memory will be reused but only when recreating the FileType.TransactionMAC file within the same application. Complete memory reuse at file level within a delegated application is possible deleting all files at once, using Format.

MF3E(H)x3 supports file creation for FileType.StandardData files not only within an application, but also at the PICC level (AID 0x0000000).

Deleting the FileType.TransactionMAC file, automatically aborts an ongoing transaction. A new transaction is started, see Section 7.14.2.2. The Transaction MAC feature gets disabled for the targeted application, see also Section 7.14.3.3.

Under active authentication, the DeleteFile requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.12.6 Memory Consumption

Memory allocation is done in 32-byte blocks, see Section 7.7.4.

The memory for files is allocated at file creation and can be computed as follows:

- General overhead: 1 block per 2 files within an application.
- Additional overhead: 1 block if one or both of the files sharing the same general overhead block has additional access rights, see Section 7.12.2.
- FileType.StandardData: $\lfloor(FileSize + 31)/32\rfloor$ blocks with *FileSize* as set with **FileSize**. If SDM or CRL is pre-enabled, 1 additional block is spent.
- FileType.BackupData: $2 \lfloor(FileSize + 31)/32\rfloor$ blocks with *FileSize* as set with **FileSize**. If CRL is pre-enabled, 1 additional block is spent.
- FileType.Value: 1 block.
- FileType.LinearRecord: $1+ \lfloor(MaxNoOfRecs * RecordSize+31)=32\rfloor$ blocks with *MaxNoOfRecs* as set with **MaxNoOfRecs** and *RecordSize* as set with **RecordSize**.
- FileType.CyclicRecord: $1+ \lfloor(MaxNoOfRecs * RecordSize+31)=32\rfloor$ blocks with *MaxNoOfRecs* as set with **MaxNoOfRecs** and *RecordSize* as set with **RecordSize**.
- FileType.TransactionMAC:
  - Transaction MAC with KeyType.AES128 FileType.TransactionMAC: 3 blocks.
  - Transaction MAC with KeyType.AES256 FileType.TransactionMAC: 4 blocks.
  - Transaction Signature with no or KeyID.AppTransactionMACKey FileType.TransactionMAC (for CommitReaderID support) : 7 blocks.
  - Transaction MAC with KeyType.AES256 FileType.TransactionMAC (for CommitReaderID support): 8 blocks.

The overhead blocks can be reused after file deletion with DeleteFile. The other file memory remains allocated until application formatting (for delegated applications) or card formatting with Format. The only exception is the overhead and file memory blocks for the FileType.TransactionMAC file. These shall all be reused when re-creating a new FileType.TransactionMAC file, as long as the newly created file fits in the already reserved memory. Note that the overhead is also not reused for other file types then.

Provided under NDA ONLY
COMPANY CONFIDENTIAL - Personal copy for:
Sergiu Salatin / Scytics GmbH
Quantum Systems
Do not copy, share or reproduce
5fc0779d-9ec-4085-901e-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.13 Data management

MF3E(H)x3 maintains user data into files of specific types as described in Section 7.12. The user can access and manage the data through functions specific to file type.

Data can be read, written or updated. Depending on the file type, data are defined as:

- raw data in FileType.StandardData and FileType.BackupData
- 4 bytes signed integer in FileType.Value files
- record in FileType.LinearRecord and FileType.CyclicRecord

For a user, the access to data is limited by the access rights set at file level as defined in Section 7.12.2 and listed in Table 46.

For all file types beside FileType.StandardData, the backup mechanism has to be considered when accessing data. The backup mechanism is defined for each file type in Section 7.12.1. Data that have been written or updated in a backup file will be visible outside the PICC only after a CommitTransaction has been successfully executed.

7.13.1 Standard and Backup data files

7.13.1.1 ReadData Command

Reading data from FileType.StandardData and FileType.BackupData files is possible with the command ReadData as defined in Section 8.8.1 and Figure 67.

There are two variants, having exact the same behavior:

- ReadData with *Cmd* 0xBD supports up to 60-byte frames and native chaining, see Section 7.5.2.2.
- ReadData with *Cmd* 0xAD supports larger FSC and ISO/IEC 14443-4 chaining, see Section 7.5.2.1.

The data to be read is defined by the file number of the targeted file, the offset in the data file where to start the reading and its size in bytes. The file number specifying the file where to read the data from is given by *FileNo* specified over 5 bits as defined in Section 7.12. *FileNo* can also indicate if a file in the secondary application is targeted via the SAI bit. If the SAI is set and no second application is selected, the command is rejected.

The position byte-wise in the data file where to start to read data is given by *Offset*. Its valid range is from 0x000000 to *FileSize - 1*. The data size to be read is given by *Length* specifying the number of bytes. If *Length* is equal to 0x000000, then the entire data file has to be read starting from the position specified by the *Offset* value. *Length* valid range is 0x000000 to *FileSize - Offset*.

The data is returned in *Data*. If the number of bytes to send to the PCD does not fit into one single frame, chaining is applied depending on the command variant as defined above, see Section 7.5.2.

For FileType.StandardData, ReadData returns data that have been written into the file even if no CommitTransaction has been successfully executed, as FileType.StandardData does not implement any backup mechanism. For FileType.BackupData, ReadData returns only data that has been written into the file and committed with a successful CommitTransaction. Before a successful commit, the old, unchanged data that was stored on the PICC will still be returned.

As listed in Table 46, ReadData is allowed only if at least one of FileAR.Read and FileAR.ReadWrite access rights associated with the targeted file is granted.

Additionally, if not authenticated, ReadData may be granted if Secure Dynamic Messaging for reading is enabled via FileAR.SDMFileRead.

If authenticated, the communication mode depends on the one from the file being accessed as specified in Section 7.5.4. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

Provided under NDA only
CONFIDENTIAL - personal copy for:
Sergio Data Systems GmbH
Guardian IT Solutions
Do not copy, share or reproduce
5fc7779d-c2c-4056-9f8-6ecc3cae421f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

MF3E(H)x3 supports this command not only within an application, but also at the PICC level (AID 0x0000000) for the FileType.StandardData files holding typically certificate data, see Section 7.10.1.2.

In case of an ongoing Transaction MAC calculation and, in case multiple applications are selected, the command is targeting a file in the application holding the FileType.TransactionMAC file, see Section 7.14.3.4, the Transaction MAC Input TMI is updated as follows:

TMI = *TMI* || *Cmd* || *FileNo* || *Offset* || *Length* || *ZeroPadding* || *Data* || *ZeroPadding*

All parameters are exactly as they appear on the command interface, except possibly *Data* and *Length*. If *Length* is set to 0x000000 in the command, meaning that the whole file is read, the actual length value is filled in with the number of bytes read for the Transaction MAC calculation. *Data* is in general the plain response data. However, if Secure Dynamic Messaging for reading is enabled, *Data* is the response data with SDM applied. In case of command chaining, the chaining overhead is ignored for the Transaction MAC computation. *Data* is the complete response data of all frames with a total byte length of *Length*.

ZeroPadding is the minimal number of zero bytes added such that the length of the TMI up to and including the *ZeroPadding* is a multiple of the cipher block size used for the Transaction MAC calculation (that is, 0 until 15 bytes are added to get a multiple of 16 bytes). Note that this padding is also added if this TMI update is not the last one before CommitTransaction.

Reading out a CRLFile, as defined in Certificate revocation is possible with this command. When reading the file content, holding the revoked CSNs, no special processing is applied. Additionally it is possible to retrieve the meta-data by setting Offset to 0xFFFFFF. In this case, Length is ignored and the parameters as detailed in Section 7.10.2.3 are returned.

The ReadData can be used to implicitly trigger an NFC Pause, as explained in Section 7.16.1.

7.13.1.2 WriteData Command

Writing data to FileType.StandardData and FileType.BackupData files is possible with the command WriteData as defined in Section 8.8.2 and Figure 68.

There are two variants, having exact the same behavior:

- WriteData with *Cmd* 0x3D supports native chaining, see Section 7.5.2.2.
- WriteData with *Cmd* 0x8D supports ISO/IEC 14443-4 chaining, see Command Size and Section 7.5.2.1.

The location of data to be written is defined by the file number of the targeted file, the offset in the data file where to start the writing and its size in bytes. The file number specifying the file where to read the records from is given by *FileNo* specified over 5 bits as defined in Section 7.12. *FileNo* can also indicate if a file in the secondary application is targeted via the SAI bit. If the SAI is set and no second application is selected, the command is rejected.

The position byte-wise in the data file where to start to write data is given by *Offset* defined on 3 bytes. Its valid range is from 0x000000 to *FileSize - 1*. The data size to be written is given by *Length* specifying the number of bytes defined on 3 bytes. *Length* valid range is 0x000001 to *FileSize - Offset*. The data is passed in *Data* and is, if needed, split in multiple frames depending on the command variant as defined above.

The FileType.StandardData does offer limited anti-tearing protection. It is ensured that the data received in a single frame is written anti-tearing protected, that is, all targeted data or none of it is updated. In case of chaining, see Section 7.5.2, the MF3E(H)x3 may buffer multiple frames and write them at once. Note that also in this case, if secure messaging applies, incomplete cryptographic blocks within a frame cannot be fully processed. Such a block will then be considered as part of the next frame.

For FileType.StandardData, data written in a file can be directly returned with ReadData, as FileType.StandardData does not implement any backup mechanism. Note especially that in case of chaining, data is already written before the integrity has been checked (CommMode.MAC or CommMode.Full). Therefore, in case of INTEGRITY_ERROR, the content of the file can be corrupted. For this reason, chained writing to

Provided under IDA only
CONFIDENTIAL - Personal copy for:
Veritis Solutions GmbH
Do not copy, share or reproduce
5fc07ed-c22-4bc-90fe-6ec3ae4f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

FileType.StandardData in CommMode.MAC or CommMode.Full can be disabled with SetConfiguration, option 0x04. Note however, that also here, the MF3E(H)x3 may buffer multiple chained frames and write them at once. In this case, the MAC is validated before the writing and all targeted data or none are updated. Therefore, this does not violate the disabled chained writing configuration.

For FileType.BackupData, data written in a file will be returned with ReadData only after a CommitTransaction has been successfully executed. Meanwhile or if CommitTransaction was not successful, the previous data content will be returned by ReadData.

As listed in Table 46, WriteData is allowed only if at least one of FileAR.Write and FileAR.ReadWrite access rights associated with the targeted file is granted.

The communication mode depends on the one from the file being accessed as specified in Section 7.5.4. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

MF3E(H)x3 supports this command not only within an application, but also at the PICC level (AID 0x0000000) for the FileType.StandardData files, see Section 7.12.1.1.

In case of an ongoing Transaction MAC calculation and, in case multiple applications are selected, the command is targeting a file in the application holding the FileType.TransactionMAC file, see Section 7.14.3.4, the Transaction MAC Input TMI is updated as follows:

$$TMI = TMI \,||\, Cmd \,||\, FileNo \,||\, Offset \,||\, Length \,||\, ZeroPadding \,||\, Data \,||\, ZeroPadding$$

All parameters are exactly as they appear on the command interface, except possibly *Data*, which is the plain command data. In case of command chaining, the chaining overhead is ignored for the Transaction MAC computation. *Data* is the complete response data of all frames with a total byte length of *Length*.

ZeroPadding is the minimal number of zero bytes added such that the length of the TMI up to and including the *ZeroPadding* is a multiple of the cipher block size used for the Transaction MAC calculation (that is, 0 until 15 bytes are added to get a multiple of 16 bytes). Note that this padding is also added if this TMI update is not the last one before CommitTransaction.

Updating a CRLFile, as defined in Certificate revocation is possible with this command. In this case, additional parameters are present as detailed in Section 7.10.2.3.3.

7.13.2 Value files

7.13.2.1 GetValue command

Reading the currently stored value from a FileType.Value file is possible with the command GetValue as defined in Section 8.8.3 and Figure 69.

The value to be retrieved is defined by the file number of the targeted value file. The file number specifying the file where to read the value from is given by *FileNo* specified over 5 bits as defined in Section 7.12.

FileNo can also indicate if a file in the secondary application is targeted via the SAI bit. If the SAI is set and no second application is selected, the command is rejected.

The value is returned in *Data* as a signed integer over 4 bytes.

FileType.Value implements a backup mechanism. Therefore updates are accumulated over the same value file but will affect the return value of GetValue only after a successful CommitTransaction has been successfully executed. Meanwhile or if CommitTransaction was not successful, the previous value will be returned by GetValue.

As listed in Table 46, then GetValue is allowed only if at least one of FileAR.Read, FileAR.Write and FileAR.ReadWrite access rights associated with the targeted file is granted or if the GetFreeValue option is set

Provided under IDEMIA - Personal copy for: COMPANY CONFIDENTIAL, Secure Data Quartet IT Solutions GmbH, Do not copy, share or replace 5fc677bd-c22-405c-90fe-66c2cae21

and if at least one of FileAR.Read, FileAR.Write and FileAR.ReadWrite access conditions is different from no access condition as specified in Section 7.12.1.3.

The communication mode depends on the one from the file being accessed as specified in Section 7.5.4. If the GetFreeValue option is activated for the targeted file as specified in GetValue and if at least one of FileAR.Read, FileAR.Write and FileAR.ReadWrite access conditions is different from no access condition then the CommMode.Plain is used for the response. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

At PICC level (AID 0x000000), the command is rejected.

In case of an ongoing Transaction MAC calculation and, in case multiple applications are selected, the command is targeting a file in the application holding the FileType.TransactionMAC file, see Section 7.14.3.4, the Transaction MAC Input TMI is updated as follows:

$$TMI = TMI \,||\, Cmd \,||\, FileNo \,||\, Value \,||\, ZeroPadding$$

All parameters are exactly as they appear on the command interface, except possibly *Value*, which is the plain 4-byte value of the response.

ZeroPadding is the minimal number of zero bytes added such that the length of the TMI up to and including the *ZeroPadding* is a multiple of the cipher block size used for the Transaction MAC calculation (that is, 0 until 15 bytes are added to get a multiple of 16 bytes). Note that this padding is also added if this TMI update is not the last one before CommitTransaction.

7.13.2.2 Credit command

Credit the value of a FileType.Value file is possible with the command Credit as defined in Section 8.8.4 and Figure 70.

The credit is defined by the file number of the targeted value file and the amount to add up. The file number specifying the file where to credit the amount is given by *FileNo* specified over 5 bits as defined in Section 7.12. *FileNo* can also indicate if a file in the secondary application is targeted via the SAI bit. If the SAI is set and no second application is selected, the command is rejected.

The credit amount is given in Data over four bytes defined as a signed integer. *Data* has to be positive. After addition, the resulting value shall lie within the limits of the value file as specified in Section 7.12.4.3. Also integer overflows are not allowed.

The value modifications of Credit, Debit and LimitedCredit operations are accumulated until CommitTransaction. Credit operations do never modify the limit available for LimitedCredit. This value can be reset to zero by issuing a LimitedCredit with value zero.

FileType.Value are implementing a backup mechanism. Therefore Credit, Debit and LimitedCredit updates are accumulated over the same value file but will affect the return value of GetValue only after a successful CommitTransaction has been successfully executed. Meanwhile or if CommitTransaction was not successful, the previous value will be returned by GetValue.

As listed in Table 46, Credit is allowed only if at least one of FileAR.ReadWrite access rights associated with the targeted file is granted.

The communication mode depends on the one from the file being accessed as specified in Section 7.5.4. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

At PICC level (AID 0x000000), the command is rejected.

In case of an ongoing Transaction MAC calculation and, in case multiple applications are selected, the command is targeting a file in the application holding the FileType.TransactionMAC file, see Section 7.14.3.4, the Transaction MAC Input TMI is updated as follows:

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Satta
Quantum Connections GmbH
Do not copy, share or reproduce
5fe37794-c22-40bc-90fc-6ecc3ca4261

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

TMI = *TMI* || *Cmd* || *FileNo* || *Value* || *ZeroPadding*

All parameters are exactly as they appear on the command interface, except possibly *Value*, which is the plain 4-byte increment value of the command.

ZeroPadding is the minimal number of zero bytes added such that the length of the TMI up to and including the *ZeroPadding* is a multiple of the cipher block size used for the Transaction MAC calculation (that is, 0 until 15 bytes are added to get a multiple of 16 bytes). Note that this padding is also added if this TMI update is not the last one before CommitTransaction.

7.13.2.3 Debit command

Decreasing the value of a FileType.Value file is possible with the command Debit as defined in Section 8.8.5 and Figure 71.

As listed in Table 46, Debit is allowed only if at least one of FileAR.Read, FileAR.Write, and FileAR.ReadWrite access rights associated with the targeted file is granted.

The debit is defined by the file number of the targeted value file and the amount to subtract. The file number specifying the file where to read the records from is given by *FileNo* specified over five bits as defined in Section 7.12. *FileNo* can also indicate if a file in the secondary application is targeted via the SAI bit. If the SAI is set and no second application is selected, the command is rejected.

The debit amount is given in *Data* over four bytes defined as a signed integer. *Data* has to be positive. After subtraction, the resulting value shall lie within the limits of the value file as specified in Section 7.12.4.3. Also integer underflows are not allowed.

A successful execution of Debit is updating the limited credit value as specified in Section 7.13.2.4.

FileType.Value are implementing a backup mechanism. Therefore Credit, Debit, and LimitedCredit updates are accumulated over the same value file but will affect the return value of GetValue only after a successful CommitTransaction has been successfully executed. Meanwhile or if CommitTransaction was not successful, the previous value will be returned by GetValue.

The communication mode depends on the one from the file being accessed as specified in Section 7.5.4. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

At PICC level (AID 0x000000), the command is rejected.

In case of an ongoing Transaction MAC calculation and, in case multiple applications are selected, the command is targeting a file in the application holding the FileType.TransactionMAC file, see Section 7.14.3.4, the Transaction MAC Input TMI is updated as follows:

TMI = *TMI* || *Cmd* || *FileNo* || *Value* || *ZeroPadding*

All parameters are exactly as they appear on the command interface, except possibly *Value*, which is the plain 4-byte decrement value of the command.

ZeroPadding is the minimal number of zero bytes added such that the length of the TMI up to and including the *ZeroPadding* is a multiple of the cipher block size used for the Transaction MAC calculation (i.e., 0 until 15 bytes are added to get a multiple of 16 bytes). Note that this padding is also added if this TMI update is not the last one before CommitTransaction.

Product under NDA only
COMPANY CONFIDENTIAL
Sergey Galata
Quantum Systems GmbH
Do not copy, share or reproduce
5fc0779d-92c-40fc-91fe-ec5ca2ffb

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.13.2.4 LimitedCredit command

Limitedly crediting the value of a FileType.Value file without having full Read/Write permissions is possible with the command LimitedCredit as defined in Section 8.8.6 and Figure 72. This feature can be enabled or disabled during value file creation.

The limited credit is defined by the file number of the targeted value file and the amount to add up. The file number specifying the file where to read the records from is given by *FileNo* specified over five bits as defined in Section 7.12. *FileNo* can also indicate if a file in the secondary application is targeted via the SAI bit. If the SAI is set and no second application is selected, the command is rejected.

The credit amount is given in *Data* over four bytes defined as a signed integer.

Data has to be positive. Data has to be smaller or equal than the limited credit limit. After addition, the resulting value shall lie within the limits of the value file as specified in Section 7.12.4.3.

The limited credit limit is defined not to be able to credit more amounts that the ones that have been debited previously. Its value is determined accumulating the debit amounts executed with Debit during the latest transaction or during the currently ongoing transaction, containing at least one debit. Therefore, debit amounts are not accumulated across multiple transactions, and the limited credit limit stays the same if a transaction does only contain Credit commands (no Debit and LimitedCredit commands). Once a LimitedCredit has been successfully executed, either during the current ongoing transaction or during a successfully committed transaction, then the limited credit limit is set to 0 not allowing anymore LimitedCredit. Therefore LimitedCredit can only be used once after a debit transaction (even if the value LimitedCredit has not been successfully committed. If that transaction is aborted, the limited credit limit is restored. Note that the limited credit limit is bound by an upper value of 0x7FFFFFFF, even if more was debited.

FileType.Value are implementing a backup mechanism. Therefore Credit, Debit and LimitedCredit updates are accumulated over the same value file but will affect the return value of GetValue only after a CommitTransaction has been successfully executed. Meanwhile or if CommitTransaction was not successful, the previous value will be returned by GetValue.

LimitedCredit is allowed if the value file enables it as defined at file creation in Section 7.12.4.3. As listed in Table 46, LimitedCredit is allowed only if at least one of FileAR.Read and FileAR.ReadWrite access rights associated with the targeted file is granted.

The communication mode depends on the one from the file being accessed as specified in Section 7.5.4. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

At PICC level (AID 0x000000), the command is rejected.

In case of an ongoing Transaction MAC calculation and, in case multiple applications are selected, the command is targeting a file in the application holding the FileType.TransactionMAC file, see Section 7.14.3.4, the Transaction MAC Input TMI is updated as follows:

$$TMI = TMI \,||\, Cmd \,||\, FileNo \,||\, Value \,||\, ZeroPadding$$

All parameters are exactly as they appear on the command interface, except possibly *Value*, which is the plain 4-byte increment value of the command.

ZeroPadding is the minimal number of zero bytes added such that the length of the TMI up to and including the *ZeroPadding* is a multiple of the cipher block size used for the Transaction MAC calculation (that is, 0 until 15 bytes are added to get a multiple of 16 bytes). Note that this padding is also added if this TMI update is not the last one before CommitTransaction.

Provided under NDA to Qualcomm Technologies Germany GmbH
COMPANY CONFIDENTIAL
Serial Number: 5fc0779a-c22c-405c-90fe-6ecc5ffa2f
Quantity: 1
Do not copy. Share or reproduce without permission.

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.13.3 Linear and Cyclic Record Files

7.13.3.1 ReadRecords Command

Reading records from FileType.LinearRecord and FileType.CyclicRecord files is possible with the command ReadRecords as defined in Section 8.8.7 and Figure 73.

There are two variants, having exact the same behavior:

- ReadRecords with *Cmd* 0xBB supports up to 60-byte frames and native chaining, see Section 7.5.2.2.
- ReadRecords with *Cmd* 0xAB supports larger FSC and ISO/IEC 14443-4 chaining, see Command Size and Section 7.5.2.1.

FileType.LinearRecord and FileType.CyclicRecord implement a backup mechanism. Therefore WriteRecord, UpdateRecord and ClearRecordFile updates over a record file will only affect the return value of ReadRecords after a successful execution of CommitTransaction. Meanwhile the previous, unchanged data returned by ReadRecords. If the transaction is aborted, for example, by AbortTransaction or unsuccessful CommitTransaction, the content of the file is not affected.

The records to be read are defined by the file number of the targeted file, the record number defining the record to start the reading from and the number of record to return. The file number specifying the file where to read the records from is given by *FileNo* specified over five bits as defined in Section 7.12.

FileNo can also indicate if a file in the secondary application is targeted via the SAI bit. If the SAI is set and no second application is selected, the command is rejected.

The record number where to start the record reading is given by *RecNo* defined on three bytes as specified in Section 7.12.1.4.1. *RecNo* valid range is from 0x000000 to *ExistingRecCount - 1*. The number of record to be read is given by *RecCount* defined on 3 bytes. If *RecCount* is equal to 0x000000 then all records from the record file has to be read from the oldest record up to and including the record referred to by *RecNo*. Else, *RecCount* valid range is 0x000001 to *ExistingRecCount - RecNo*. The records are returned in *Data* in the chronological order returning the oldest record as defined in Section 7.12.1.4.1 first. If the number of bytes to send to the PCD does not fit into one single frame, chaining is applied depending on the command variant as defined above.

An attempt to read more records than have been written so far, will be rejected. This includes attempt to read from empty record files, that is, immediately after file creation or after committed clearance with ClearRecordFile. Note also that the maximum number of records of a FileType.CyclicRecord file is one less than specified at creation.

As listed in Table 46, ReadRecords is allowed only if at least one of FileAR.Read and FileAR.ReadWrite access rights associated with the targeted file is granted.

The communication mode depends on the one from the file being accessed as specified in Section 7.5.4. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

At PICC level (AID 0x000000), the command is rejected.

In case of an ongoing Transaction MAC calculation and, in case multiple application are selected, the command is targeting a file in the application holding the FileType.TransactionMAC file, see Section 7.14.3.4, the Transaction MAC Input TMI is updated as follows:

$$TMI = TMI \,||\, Cmd \,||\, FileNo \,||\, RecNo \,||\, RecCount \,||\, ZeroPadding \,||\, Data \,||\, ZeroPadding$$

All parameters are exactly as they appear on the command interface, except possibly *Data* and *RecCount*. If *RecCount* is set to 0x000000 in the command, meaning that all records are read (starting from the *RecNo*), the actual length value is filled in with the number of records read for the Transaction MAC computation. In

Provided under NXP Semiconductors GmbH CONFIDENTIAL - Personal copy for: Sergio Salata Catalog IT Solutions GmbH Do not copy, share or reproduce 5fc079dc22c405e0b4ec3cae42f

case of command chaining, the chaining overhead is ignored for the Transaction MAC computation. *Data* is the complete response data of all records read.

ZeroPadding is the minimal number of zero bytes added such that the length of the TMI up to and including the *ZeroPadding* is a multiple of the cipher block size used for the Transaction MAC calculation (that is, 0 until 15 bytes are added to get a multiple of 16 bytes). Note that this padding is also added if this TMI update is not the last one before CommitTransaction.

7.13.3.2 WriteRecord Command

Writing a record to FileType.LinearRecord and FileType.CyclicRecord files is possible with the command WriteRecord as defined in Section 8.8.8 and Figure 74.

There are two variants, having exact the same behavior:

- WriteRecord with *Cmd* 0x3B supports native chaining, see Section 7.5.2.2.
- WriteRecord with *Cmd* 0x8B supports ISO/IEC 14443-4 chaining, see Command Size and Section 7.5.2.1.

FileType.LinearRecord and FileType.CyclicRecord implement a backup mechanism. Therefore WriteRecord, UpdateRecord and ClearRecordFile updates over a record file will only affect the return value of ReadRecords after a successful execution of CommitTransaction. Meanwhile or if CommitTransaction was not successful, the previous records will be returned by ReadRecords.

For FileType.LinearRecord file, the first WriteRecord in a transaction appends a new record at the end of the file. It is then not possible to execute a WriteRecord if the file is already full. For FileType.CyclicRecord file, the first WriteRecord in a transaction either appends a new record if not all records have been written or erases and overwrites the oldest record if the existing number of records equals the maximum number of records of the file. Note that the maximum number of records of a FileType.CyclicRecord file is one less than specified at creation.

In the current transaction if a UpdateRecord or ClearRecordFile has already been successfully done over one record file then it is not possible to execute a WriteRecord on the same record file. In the current transaction, a record file can have the same record written multiple times with WriteRecord.

The record data to be written is defined by the file number of the targeted file, the offset in bytes within the new record to start the writing, the number of bytes to write and the data. The file number specifying the file where to read the records from is given by *FileNo* specified over 5 bits as defined in Section 7.12.

FileNo can also indicate if a file in the secondary application is targeted via the SAI bit. If the SAI is set and no second application is selected, the command is rejected.

The position byte-wise within the record where to write the data is given by *Offset* defined on three bytes. *Offset* valid range is from 0x000000 to *RecordSize-1*. The number of bytes to be written is given by *Length* defined on three bytes. *Length* valid range is from 0x000001 to *RecordSize - Offset*.

An attempt to write outside record boundaries will be rejected. If it is the first write operation to the record in the transaction, any unwritten bytes will be set to zero.

As listed in Table 46, WriteRecord is allowed only if at least one of FileAR.Read and FileAR.ReadWrite access rights associated with the targeted file is granted.

The communication mode depends on the one from the file being accessed as specified in Section 7.5.4. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

At PICC level (AID 0x000000), the command is rejected.

In case of an ongoing Transaction MAC calculation and, in case multiple applications are selected, the command is targeting a file in the application holding the FileType.TransactionMAC file, see Section 7.14.3.4, the Transaction MAC Input TMI is updated as follows:

$$TMI = TMI \,||\, Cmd \,||\, FileNo \,||\, Offset \,||\, Length \,||\, ZeroPadding \,||\, Data \,||\, ZeroPadding$$

Provided under NDA ONLY
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Santa Quintana IT Solutions GmbH
Do not copy, share or reproduce
55c07f9d-c22c-455c-96fe-2ec0ccae2251

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

All parameters are exactly as they appear on the command interface, except possibly *Data*, which is the plain command data. In case of command chaining, the chaining overhead is ignored for the Transaction MAC computation. *Data* is the complete command data of all frames with a total byte length of *Length*.

ZeroPadding is the minimal number of zero bytes added such that the length of the TMI up to and including the *ZeroPadding* is a multiple of the cipher block size used for the Transaction MAC calculation (that is, 0 until 15 bytes are added to get a multiple of 16 bytes). Note that this padding is also added if this TMI update is not the last one before CommitTransaction.

7.13.3.3 UpdateRecord Command

The UpdateRecord updates a record within a FileType.LinearRecord or a FileType.CyclicRecord file and is defined in Section 8.8.9 and Figure 75.

There are two variants, having exact the same behavior:

- UpdateRecord with *Cmd* 0xDB supports native chaining, see Section 7.5.2.2.
- UpdateRecord with *Cmd* 0xBA supports ISO/IEC 14443-4 chaining, see Command Size and Section 7.5.2.1.

FileType.LinearRecord and FileType.CyclicRecord implement a backup mechanism. Therefore WriteRecord, UpdateRecord and ClearRecordFile updates over a record file will only affect the return value of ReadRecords after a successful execution of CommitTransaction.

In the current transaction if a UpdateRecord has already been successfully done over the targeted record file but on a different record, then it is not possible to execute a UpdateRecord on the same record file. It is possible to have one single current record updated multiple times with UpdateRecord within the same transaction.

In the current transaction if a WriteRecord or ClearRecordFile has already been successfully done over one record file then it is not possible to execute a UpdateRecord on the same record file.

The record data to be written is defined by the file number of the targeted file, the record number, the offset in bytes within the record to start the updates, the number of bytes to update and the data. The file number specifying the file where to read the records from is given by *FileNo* specified over five bits as defined in Section 7.12. *FileNo* can also indicate if a file in the secondary application is targeted via the SAI bit. If the SAI is set and no second application is selected, the command is rejected.

The record number of the record to update is given by *RecNo* defined on three bytes as specified in Section 7.13.3. *RecNo* valid range is from 0x000000 to *ExistingRecCount - 1*. The position byte-wise within the record where to write the data is given by UpdateRecord. Offset defined on three bytes.

Offset valid range is from 0x000000 to *RecordSize - 1*. The number of bytes to be written is given by *Length* defined on 3 bytes. *Length* valid range is from 0x000000 to *RecordSize - Offset*.

An attempt to update a record that has not been written before or to update outside record boundaries will be rejected. For a FileType.LinearRecord file, one additional record is required to temporarily store the updated record before the CommitTransaction is executed. Therefore, one additional record should be created on top of the desire number of records if the UpdateRecord command is to be used. This additional record should not be written and the record file should never be full. Once the file is full, that is, the maximum number of records have been written, it is not possible to update any records anymore.

As listed in Table 46, UpdateRecord is allowed only if at least one of FileAR.ReadWrite access rights associated with the targeted file is granted.

The command ClearRecordFile requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

At PICC level (AID 0x000000), the command is rejected.

Provided under NDA CONFIDENTIAL Personal copy for: Sergi Quintana Solutions Do not copy, share or reproduce 5fc07f9d-d2c4-455-90fe-6cc5ca42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

In case of an ongoing Transaction MAC calculation and, in case multiple applications are selected, the command is targeting a file in the application holding the FileType.TransactionMAC file, see Section 7.14.3.4, the Transaction MAC Input TMI is updated as follows:

TMI = *TMI* || *Cmd* || *FileNo* || *RecNo* || *Offset* || *Length* || *ZeroPadding* || *Data* || *ZeroPadding*

All parameters are exactly as they appear on the command interface, except possibly *Data*, which is the plain command data. In case of command chaining, the chaining overhead is ignored for the Transaction MAC computation. *Data* is the complete command data of all frames with a total byte length of *Length*.

ZeroPadding is the minimal number of zero bytes added such that the length of the TMI up to and including the *ZeroPadding* is a multiple of the cipher block size used for the Transaction MAC calculation (that is, 0 until 15 bytes are added to get a multiple of 16 bytes). Note that this padding is also added if this TMI update is not the last one before CommitTransaction.

7.13.3.4 ClearRecordFile Command

Resetting FileType.LinearRecord and FileType.CyclicRecord files to the empty state is possible with the command ClearRecordFile as defined in Section 8.8.10 and Figure 76.

FileType.LinearRecord and FileType.CyclicRecord implement a backup mechanism. Therefore WriteRecord, UpdateRecord and ClearRecordFile updates over a record file will only affect the return value of ReadRecords after a successful execution of CommitTransaction. Meanwhile or if CommitTransaction was not successful, the previous records will be returned by ReadRecords.

In the current transaction if a ClearRecordFile has already been successfully done over one record file then it is not possible to execute a WriteRecord or a UpdateRecord on the same record file. But ReadRecords still returns the previous records from the file. Once a CommitTransaction has been successfully executed, the file is empty again: then WriteRecord is again possible and ReadRecords and UpdateRecord will fail due to empty file.

The record file to be cleared is defined by the file number of the targeted file. The file number specifying the file where to read the records from is given by *FileNo* specified over five bits as defined in Section 7.12. *FileNo* can also indicate if a file in the secondary application is targeted via the SAI bit.

If the SAI is set and no second application is selected, the command is rejected.

As listed in Table 46, ClearRecordFile is allowed only if at least one of FileAR.ReadWrite access rights associated with the targeted file is granted.

Under active authentication, the command ClearRecordFile requires CommMode.MAC. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

At PICC level (AID 0x000000), the command is rejected.

In case of an ongoing Transaction MAC calculation and, in case multiple applications are selected, the command is targeting a file in the application holding the FileType.TransactionMAC file, see Section 7.14.3.4, the Transaction MAC Input TMI is updated as follows:

TMI = *TMI* || *Cmd* || *FileNo* || *ZeroPadding*

All parameters are exactly as they appear on the command interface.

ZeroPadding is the minimal number of zero bytes added such that the length of the TMI up to and including the *ZeroPadding* is a multiple of the cipher block size used for the Transaction MAC calculation (that is, 0 until 15 bytes are added to get a multiple of 16 bytes). Note that this padding is also added if this TMI update is not the last one before CommitTransaction.

Provided under NDA to COMPANY CONFIDENTIAL - Personal copy for: Sebastian Sauter Quantum Solutions GmbH Do not copy, share or reproduce 5fc07704-c22c-4956-9012-66c3ce4af1

7.13.4 Transaction MAC files

7.13.4.1 ReadData Command

Reading data from FileType.TransactionMAC files is possible with the command ReadData as defined in Section 8.8.1.

The behavior is exactly the same as for FileType.BackupData files of 12 bytes, see Section 7.13.1.1.

There is one exception: the Transaction MAC Input TMI of the ongoing Transaction MAC calculation is not updated when reading a FileType.TransactionMAC file.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.14 Transaction management

MF3E(H)x3 supports the grouping of data management operations into transactions. This allows for committing or aborting all previous write access to files with integrated backup mechanisms within the selected application(s) via a single atomic operation.

How a transaction can be initiated and concluded, is detailed in Section 7.14.1 and Section 7.14.2.

On top of this, MF3E(H)x3 supports a feature which allows proofing transaction execution toward a third party. This feature can be used if, for example, a merchant needs to be reimbursed by a backend for the transaction they executed. This way the backend is protected against the fraudulent merchant scenario.

This feature comes in two variants:

- Transaction MAC: based on symmetric crypto, equivalent to the feature supported by MIFARE DESFire. Specified in section Section 7.14.3.
- Transaction Signature: based on asymmetric crypto, newly introduced in MF3E(H)x3. Specified in Section 7.14.4.

Note that the use cases for both variants are very similar, but the Transaction Signature variant has the advantage that the verification of the transaction credential can happen based on a public key. Therefore, this can not only be done in the backend, but also already at the merchant or reader terminal. This may allow for optimizing the amount of asymmetric crypto operations required during a transaction, by only authenticating the card at the end of the transaction via this signature. See subsection Section 7.14.4.6, for some more details and considerations.

7.14.1 Transaction initiation

MF3E(H)x3 does not provide a separate command for initiating a transaction. Instead, a transaction is started by selecting one or two applications with SelectApplication, see Section 7.11.3 or ISOSelectFile. Note that if two applications are selected, the transaction can affect files in both selected applications.

Next to this, a new transaction is also automatically initiated after conclusion of a previous transaction, see Section 7.14.2.

Note that a transaction is not affected by the execution of an authentication. One can have several authetncations during the course of a single transaction.

7.14.2 Transaction conclusion

A transaction can be concluded by successfully committing all write operations on backup files, as detailed in Section 7.14.2.1, or by abortion which will cancel all changes, as detailed in Section 7.14.2.2. For both operations, MF3E(H)x3 supports a specific command, but transaction abortion can also be triggered by other commands.

Note that some of the parameters and actions triggered by these commands are related with the Transaction MAC feature, explained in Section 7.14.3. For a full understanding, the description in that section should be referred to.

7.14.2.1 CommitTransaction command

Validating all previous write access to back up files within the currently selected application(s) is possible with the command CommitTransaction as defined in Section 8.9.1 and Figure 77.

The CommitTransaction takes an optional parameter *Option*, which indicates if the TMC and TMV or TSV of the calculated Transaction MAC/Signature, see Section 7.14.3 /Section 7.14.4, are to be sent with the response. If this is requested while the selected application(s) do not support Transaction MAC/Signature calculation, that is, no FileType.TransactionMAC file is present, the command is rejected.

Provided under NDA only - Personal copy for: COMMENT CONFIDENTIAL Sergii Salata Quantagit Solutions GmbH Do not copy, share or reproduce 5f0d7f0d-c2ac-495c-90fe-6ecc3ca42f

The CommitTransaction validates all preceding write access of the ongoing transaction to files with integrated backup mechanisms:

- FileType.BackupData
- FileType.Value
- Linear and Cyclic Record Files
- FileType.CyclicRecord

If the Transaction MAC or Siganture feature is enabled, as one of the selected applications contains a FileType.TransactionMAC file, the FileType.TransactionMAC file is updated, even if TMI is the empty string. The updated file will hold the calculated TMV, as defined in Section 7.14.3.2.4 and the increased TMC which was used for the calculation of KeyID.SesTMMACKey.

In case of Transaction Signature, the calculated TSV, as defined in Section 7.14.4.2.4 and the increased TMC.

If TMC reaches the TMCLimit, see Section 7.14.3.2.2, the application remains selected, but with limited functionality: CommitTransaction will be rejected, that is, no further transactions can be executed.

All these file updates are processed as a single atomic operation. In case of tearing, either all or none of the changes are applied to the memory.

After execution of this command a new transaction is started.

The command is rejected if no application has been selected, that is, the PICC level (AID 0x0000000) is currently selected.

An error code is also returned if no write access has been done to any of the backup file types and, in case the application is protected by the Transaction MAC feature (see Section 7.14.3), no Transaction MAC calculation update has been done. This means no command that is included in the Transaction MAC calculation has been executed, that is, the TMI is still the empty string. This is also the case if two applications are selected when read operations have only been executed on the application without FileType.TransactionMAC. Note that if Transaction MAC is enabled, successful execution of CommitTransaction updating the FileType.TransactionMAC is possible without any write operation, as read operations are also included in the Transaction MAC calculation.

If CommitReaderID is required according to access right configuration of the FileType.TransactionMAC, this command will be rejected if no *ReaderID* has been committed during the ongoing transaction, that is, if TMRICur is still the empty string. If a ReaderID was committed, this value TMRICur is stored in non-volatile memory as TMRIPrev as part of the atomic operation.

The CommitTransaction requires CommMode.MAC if there is an active authentication. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

The communication mode of the response, if the optional response data TMC and TMV are returned, depends on the FileType.TransactionMAC file. Note that there is not necessarily an authentication with one of the FileType.TransactionMAC file access keys required. If there is no active authentication, TMC and TMV are returned in plain.

The CommitTransaction is typically the last command of a transaction before deselecting the virtual card with the ISO/IEC 14443-4 Deselect command, or before proceeding with another application with SelectApplication.

Note that TMC and TMV can also be read out encrypted, if preferred. In this case the FileType.TransactionMAC file should be configured for CommMode.Full. One can then use ReadData to retrieve this information, instead of requesting it within the response of CommitTransaction.

Provided under NDA only
COMPANY CONFIDENTIAL - personal copy for:
Seitz Salzgitter GmbH
Quantas IT Solutions GmbH
Do not copy, share or reproduce
5fc07793-92c4-405c-90fc-63cae421

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.14.2.2 AbortTransaction command

A transaction can be aborted by explicitly issuing a AbortTransaction.

Next to this any ongoing transaction is also automatically aborted after:

- unsuccessful execution of a command, see Section 7.5.3.
- selecting an application with SelectApplication or ISOSelectFile, even if the same application is reselected.
- deleting the selected application with DeleteApplication
- formatting the selected application with Format
- creating/deleting the FileType.TransactionMAC file with CreateTransactionMACFile or DeleteFile
- rolling the key set with RollKeySet
- enabling TMCLimit with ChangeFileSettings

This has the same effects as successful execution of AbortTransaction detailed below. A new transaction is started, except if the PICC level (AID 0x000000) is selected afterward (for example, if the selected application is deleted).

Invalidating all previous write access to back up files within the currently selected application(s) is possible with the command AbortTransaction as defined in Section 8.9.2 and Figure 78.

This command takes no parameter.

The AbortTransaction invalidates all preceding write access of the ongoing transaction to files with integrated backup mechanisms:

- FileType.BackupData
- FileType.Value
- Linear and Cyclic Record Files
- FileType.CyclicRecord

If a Transaction MAC or Signature is calculated, see Section 7.14.3, the ongoing calculation is also aborted and a new calculation is initiated, see Section 7.14.3.4.2.

The AbortTransaction does not affect the authentication status. After executing this command, a new transaction is started.

The command is rejected if no application has been selected, that is, the PICC level (AID 0x0000000) is currently selected.

An error code is also returned if no write access has been done to any of the backup file types and, if the application is protected by the Transaction MAC / Signature feature, the TMI is still the empty string. This is similar as for CommitTransaction.

The AbortTransaction requires CommMode.MAC if there is an active authentication. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

This is useful to cancel a transaction without the need for re-authentication to the PICC or without a reset of the RF field and a complete new ISO/IEC 14443 activation and possibly VC Selection procedure, which would have the same effect.

7.14.3 Transaction MAC

The Transaction MAC feature helps preventing fraudulent merchant attacks. It allows a merchant operating a point-of-sale terminal to prove that the transactions he executed with customers are genuine toward the card issuers or application provider's backend.

This is done by letting the card generate a Transaction MAC over the transaction with a key shared only by the card and the card issuers (or application provider's) backend. This key is called the

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

KeyID.AppTransactionMACKey. As this key will not be present in the merchants terminal, it is not possible for the merchant to forge MACs toward the backend. Note that this implies that (pre-) personalization of the card is done by the card issuer and not at the merchant's location.

Note that this section describes the variant of the feature based on symmetric crypto, as is also supported by MIFARE DESFire. The Transaction Signature variant based on asymmetric crypto is described in Section 7.14.4.

MAC generation will also involve a Transaction MAC Counter maintained by the card, which allows the backend to detect replay by the merchant. In addition this counter allows the backend to detect missing transactions, which allows detecting fraud where the merchant has increased the balance without reporting the transaction. Note that a counter also allows for easier detection mechanisms of replay than keeping logs with fingerprints of every transaction.

Fraud detection alone might not be sufficient if one cannot point to the fraudulent merchant. Therefore, an additional configuration option requires that the merchants terminal commits its identity during the transaction. This requires support of a SAM (Secure Access Module) which allows a secure commit of an ID that is then linked with the merchant. The goal of this is that the merchant cannot forge the committed ID. This ID is called ReaderID. For this feature, a key is assigned which known by both the card and the SAM: KeyID.AppCommitReaderIDKey. At the merchant's site, this key shall not leave the SAM.

The card will support linking ReaderIDs of merchants by storing the committed ReaderID and returning the ReaderID of the previous transaction. This way the backend can detect which merchant did not report missing transactions.

7.14.3.1 General overview

The Transaction MAC is an application level feature. It can be activated per application by creating a specific file: KeyID.AppTransactionMACKey. When creating this file, one also specifies the KeyID.AppTransactionMACKey, see Section 7.12.4.6. Additionally, one can optionally also require the commitment of a ReaderID, by assigning a KeyID.AppCommitReaderIDKey for this, see Section 7.12.1.5.1.

Once the Transaction MAC feature is activated, the Transaction MAC computation consists of the following steps:

1. **Initialization:** before performing the first command that needs a Transaction MAC computation update (see below), the Transaction MAC Session Keys KeyID.SesTMMACKey and KeyID.SesTMENCKey are derived from the KeyID.AppTransactionMACKey using the next Transaction MAC Counter value, see also Section 7.14.3.2.1.
2. **Computation updates:** during the transaction, the Transaction MAC is computed using KeyID.SesTMMACKey over the following commands:
   - ReadData, GetValue, **and** ReadRecords**:** these commands are not necessarily executed during a transaction, especially for FileType.Value and FileType.LinearRecord or FileType.CyclicRecord files. However, there might be use cases where a backend also needs the current value of a file to be authenticated. For example for FileType.BackupData files, as the data written might not be sufficient to derive the amount of an increment or decrement operation. For that you also must know the original data. This is made possible by including these commands in the ongoing Transaction MAC computation if they are executed. On a file update command, the current value of the file is not included in the Transaction MAC computation, such that a user is not forced to read the data to report toward the backend if this is not required by the system.
   - WriteData, Credit, Debit, LimitedCredit, WriteRecord, UpdateRecord, **and** ClearRecordFile**:** these commands are essential to be authenticated toward the backend, as the merchant must proof execution of them toward the backend to claim reimbursement. So these are included in the Transaction MAC computation.

Provided under NXP CONFIDENTIAL Qualified Solutions GmbH Certificate 974412 Do not copy, share or reproduce 5fc07f79-c2f2-405c-90fe-8ccc3cee4201

- CommitReaderID: the committed ReaderID is stored on the card and the ReaderID of the previous transaction is returned encrypted with the KeyID.SesTMENCKey. This is also included in the ongoing Transaction MAC computation.

3. **Finalization:** on CommitTransaction, the Transaction MAC is finalized. The update of the manipulated backup files and the computed Transaction MAC, the related counter, and (optionally) the committed ReaderID, are one atomic operation (as the files should not be updated without the MAC being available and vice versa). The computed Transaction MAC and the related counter are either provided with the response of CommitTransaction or can be read afterward using ReadData on the FileType.TransactionMAC file.

7.14.3.2 Cryptographic and other concepts

7.14.3.2.1 Transaction MAC Counter

Each Transaction MAC calculation is associated with a Transaction MAC Counter (TMC). This is a four byte unsigned integer. On the interface or in cryptographic calculations, the TMC is represented LSB first. The TMC is stored in the FileType.TransactionMAC file.

The current value inside FileType.TransactionMAC, as can be read out, is called TMC. The TMC added by one must serve as input for the Transaction MAC Session Key generation, see Section 7.14.3.2.3. The FileType.TransactionMAC file will only reflect this updated value after a successful transaction, concluded with CommitTransaction.

If the TMC equals the value 0xFFFFFFFF with TMCLimit disabled, this is equivalent to TMCLimit being reached when enabled, see next section.

7.14.3.2.2 Transaction MAC Counter Limit

The number of transactions that can be executed within an application can be limited by setting a Transaction MAC Counter Limit (TMCLimit). This is an unsigned integer of four bytes. On the interface or in cryptographic calculations, the TMC is represented LSB first.

At delivery, the TMCLimit is disabled. This is equivalent to holding the maximum value (0xFFFFFFFF). The TMCLimit can be enabled by setting a customized value with ChangeFileSettings. It can be retrieved with GetFileSettings.

Once the stored TMC equals the TMCLimit, CommitTransaction will be rejected, that is, no further transactions can be executed. This means the application can still be selected, but with limited functionality. ISOSelectFile will be responded with ISO6283.

If the TMCLimit gets disabled with ChangeFileSettings, this is also equivalent to putting it to the maximum value: 0xFFFFFFFF. Also when the FileType.TransactionMAC file gets re-created with CreateTransactionMACFile, the feature is disabled and TMCLimit is set to the maximum value.

Note that in practice this means that when configuring TMCLimit to, for example, 5 when the TMC is still zero, no Transaction MAC protected transactions have been executed, five transactions can successfully be executed, that is, for TMC holding the values 1 until 5.

When the TMCLimit is reached, reading the FileType.TransactionMAC file allows to retrieve TMC and TMV still. CommitTransaction will be rejected, that is, no further transactions can be executed. Note that this is slightly different from pure MIFARE DESFire Light implementations, where all data management commands, except ReadData targeting the FileType.TransactionMAC file, will be rejected. However, it is possible to increase the TMCLimit with ChangeFileSettings, or disable the Transaction MAC feature overall by deleting the FileType.TransactionMAC file with DeleteFile, and in this way make the application fully functional again.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Certificate Solutions GmbH
Quality Assurance or reproduce
Do not copy, share or reproduce
5fc07bd-c2c4-405c-90fe-6ecc3ca421

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.14.3.2.3 Transaction MAC Session Keys

Out of the KeyID.AppTransactionMACKey, two session keys are generated:

- KeyID.SesTMMACKey or $K_{SesTMMAC}$ for computing the Transaction MAC Value
- KeyID.SesTMENCKey or $K_{SesTMENC}$ for encrypting the committed ReaderID (if used)

The Transaction MAC Session Keys are derived using the following algorithms.

The session key generation is according to NIST SP 800-108 [19] in counter mode.

The pseudo-random function PRF(key; message) applied during the key generation is the CMAC algorithm described in NIST Special Publication 800-38B [18]. The key derivation key is the KeyID.AppTransactionMACKey which is always of KeyType.AES.

The input data is constructed using the following fields as defined by [19]. Note that NIST SP 800-108 allows defining a different order than proposed by the standard as long as it is unambiguously defined.

- a 1-byte label, distinguishing the purpose of the key:
  - 0x5A for MACing
  - 0xA5 for encryption
- a 2-byte counter
  - 0x0001 for 128-bit keys
  - 0x0001 and 0x0002 for 256-bit keys
- a 2-byte length
  - 0x0080 for 128-bit keys
  - 0x0100 for 256-bit keys
- a 11-byte context, constructed using the 4-byte TMC+1 and the UID, followed by zero-byte padding if needed

KeyType.AES128

Two session vectors SV_x are derived as follows:

$$SV1 = 0x5A \,||\, 0x00 \,||\, 0x01 \,||\, 0x00 \,||\, 0x80 \,||\, (TMC+1) \,||\, UID[\,||\, 0x00..0x00]$$
$$SV2 = 0xA5 \,||\, 0x00 \,||\, 0x01 \,||\, 0x00 \,||\, 0x80 \,||\, (TMC+1) \,||\, UID[\,||\, 0x00..0x00]$$

Padding with zeros is done up to a multiple of 16 bytes. So in case of 7-byte *UID*, no padding is added.

Then, the 16-byte session keys KeyID.SesTMMACKey and KeyID.SesTMENCKey are constructed as follows:

$$K_{SesTMMAC} = PRF(K_{AppTransactionMAC}, SV1)$$
$$K_{SesTMENC} = PRF(K_{AppTransactionMAC}, SV2)$$

KeyType.AES256

Four session vectors SV_{xy} are derived as follows:

$$SV1a = 0x5A \,||\, 0x00 \,||\, 0x01 \,||\, 0x01 \,||\, 0x00 \,||\, (TMC+1) \,||\, UID[\,||\, 0x00..0x00]$$
$$SV1b = 0x5A \,||\, 0x00 \,||\, 0x02 \,||\, 0x01 \,||\, 0x00 \,||\, (TMC+1) \,||\, UID[\,||\, 0x00..0x00]$$
$$SV2a = 0xA5 \,||\, 0x00 \,||\, 0x01 \,||\, 0x01 \,||\, 0x00 \,||\, (TMC+1) \,||\, UID[\,||\, 0x00..0x00]$$
$$SV2b = 0xA5 \,||\, 0x00 \,||\, 0x02 \,||\, 0x01 \,||\, 0x00 \,||\, (TMC+1) \,||\, UID[\,||\, 0x00..0x00]$$

Padding with zeros is done up to a multiple of 16 bytes. So in case of 7-byte *UID*, no padding is added.

Then, the 32-byte session keys KeyID.SesTMMACKey and KeyID.SesTMENCKey are constructed as follows:

$$K_{SesTMMAC} = PRF(K_{AppTransactionMAC}, SV1a) \,||\, PRF(K_{AppTransactionMAC}, SV1b)$$
$$K_{SesTMENC} = PRF(K_{AppTransactionMAC}, SV2a) \,||\, PRF(K_{AppTransactionMAC}, SV2b)$$

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.14.3.2.4 Transaction MAC Value and Input

The 8-byte Transaction MAC Value (TMV) is computed over the Transaction MAC Input (TMI). This input depends on the commands executed during the transaction, see Section 7.14.3.4. The applied key is KeyID.SesTMMACKey, defined in Section 7.14.3.2.3.

A different notation than the one for Secure Messaging MACs is used: $MACt_{TM}(key, message)$ is used to denote the CMAC operation including truncation. $MAC_{TM}(key, message)$ denotes the CMAC result before truncation. Note that, though similar algorithm as for Secure Messaging is used, this MAC calculation is unrelated with the secure messaging itself as a different key is applied.

The TMV is calculated as follows:

$$TMV = MACt_{TM}(K_{SesTMMAC}, TMI)$$

using the MAC algorithm of the EV2 Secure Messaging with zero byte IV, see Section 7.5.10.3.

7.14.3.2.5 Transaction MAC ReaderID and its encryption

A Transaction MAC ReaderID is 16 bytes. If ReaderID commitment is enabled, see Section 7.14.3.4.4, two ReaderIDs are maintained by the card.

- TMRICur: the current ReaderID of the ongoing transaction. It is set with CommitReaderID and typically kept in volatile memory as it does not need to survive PICC power off.
- TMRIPrev: the ReaderID of the latest successful transaction. On successful execution of CommitTransaction, TMRICur is stored on the PICC as TMRIPrev. This is to be kept in non-volatile memory as it survives PICC power off.

During the next transaction, this TMRIPrev is returned encrypted using KeyID.SesTMENCKey. This is done via the *EncTMRI* parameter in the response of the CommitReaderID:

$$EncTMRI = E_{TM}(K_{SesTMENC}, TMRIPrev)$$

using the AES block cipher according to the CBC mode of NIST SP800-38A [17] without adding any padding. The zero byte IV is applied, that is, 128 bits of 0.

The initial TMRIPrev, as configured during creation of the FileType.TransactionMAC, is set to all zero bytes, see Section 7.12.4.6.

The exact specification of the TMRI is out of scope for this document, but an example can be the 7-byte SAM UID with padding.

As KeyID.SesTMENCKey is different for each transaction (due to inclusion of TMC in the session key generation), a constant IV does not leak any information.

A different notation than the one for Secure Messaging encryption is used: $E_{TM}(key, message)$. Note that, though the same algorithm as the EV2 Secure Messaging is used, this encryption is unrelated with the secure messaging itself as a different key is applied.

7.14.3.3 Transaction MAC enabling

The Transaction MAC feature is an application level feature and is therefore enabled and disabled per application.

The Transaction MAC feature is enabled for an application by creating a FileType.TransactionMAC file with CreateTransactionMACFile, see Section 7.12.4.6. An application can hold only one FileType.TransactionMAC file.

Once enabled, the Transaction MAC feature can be disabled for an application by either deleting the FileType.TransactionMAC file with DeleteFile, or by deleting all files in the application with Format.

Provided under NDA only - Personal copy for: COMPANY CONFIDENTIAL Sergi Satta Quantum IT Solutions GmbH Do not copy, share or reproduce 5fa07779d-c22c-495c-90fe-6ecc3cae42f1

As soon as the Transaction MAC feature is enabled for a selected application, the Transaction MAC will be calculated, as specified in the next section, for every transaction within the currently selected application, starting with the next transaction.

MF3E(H)x3 supports addition Transaction MAC related configuration options which can be set via ChangeFileSettings:

- enabling a Transaction MAC counter limit TMCLimit.

The KeyID.AppTransactionMACKey is not an KeyID.AppKey. Instead it is fully specified and set by the CreateTransactionMACFile. Therefore, if this key needs to be changed, the FileType.TransactionMAC file needs to be deleted and re-created. Notice that by doing this, the TMC is also reset to zero.

7.14.3.4 Transaction MAC calculation

7.14.3.4.1 Transaction MAC state diagram

The Transaction MAC calculation is described by the Transaction MAC state diagram given in Figure 18. The different actions related with the shown commands are detailed in the following subsections section. Commands not shown in the figure do not influence the Transaction MAC computation.

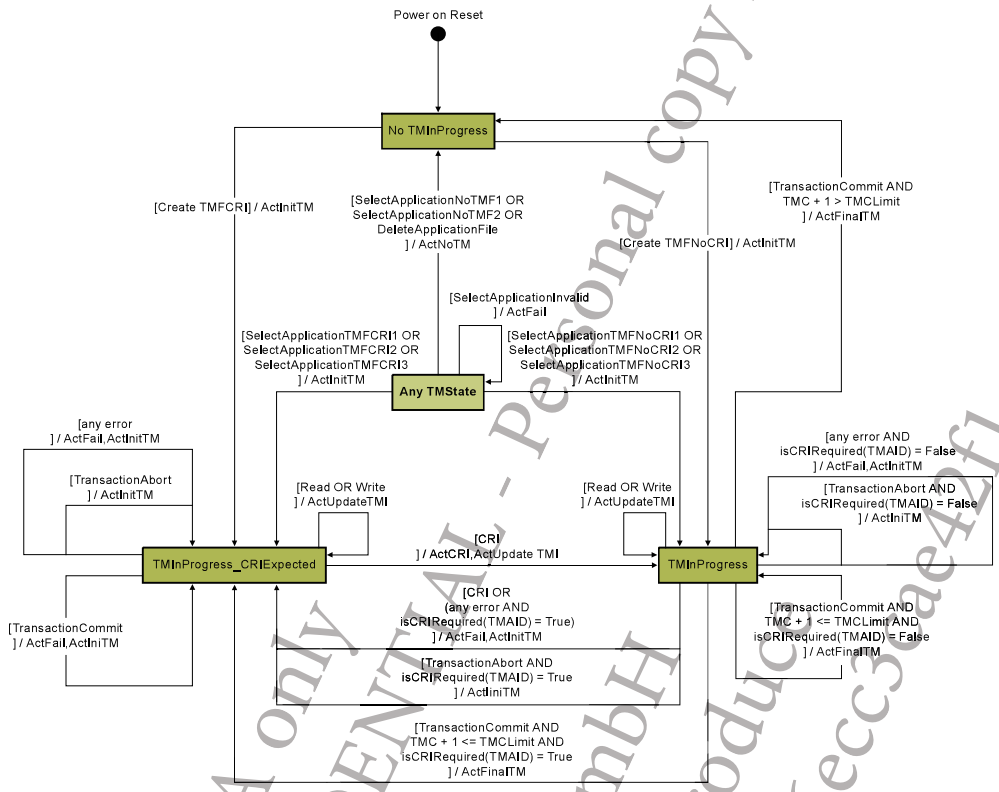
The state Any TMState is a placeholder for any of the three other Transaction MAC States. This is done to reduce the number of arrows.

Note that the action description is not normative. Certain actions or subactions could be implemented at other points in time. This is allowed as long as there are no explicit requirements and the overall behavior over the command interface remains the same. For example, session key generation could be delayed until the first Read or Write operation occurs.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.



Actions

ActInitTM:

Reset TMI to the empty string.
Reset TMRICur to the empty string.
Set TMAID to the selected application containing a File Type, TransactionMAC.
Compute session keys from KeyID.AppTransactionMACKey with TMC+1.

ActNoTM:

Reset TMI, TMRICur and TMAID.
Destroy session keys.

ActUpdateTMI:

Update TMI with the cmd related string.

ActCRI:

[If authenticated] Set TMRICur to the received ReaderID.
[If authenticated] Return the ReaderID of the previous transaction TMRIPrev encrypted.
[If not authenticated] No specific ActCRI action.

ActFinalTM:

Compute TMV.
Write anti-teared:
- TMC +1 || TMV into FileType. TransactionMAC
- [if TMRICur != empty string] TMRIPrev = TMRICur

ActInitTM if written TMC <= TMCLimit

ActFail:

Send the appropriate error return code.

Legend

TMAID: the currently selected application holding a FileType. TransactionMAC

Read:
Cmd.ReadData
Cmd.GetValue
Cmd.ReadRecords

CRI:
Cmd.CommitReaderID

Write:
Cmd.WriteData
Cmd.Credit
Cmd.Debit
Cmd.LimitedCredit
Cmd.WriteRecord
Cmd.UpdateRecord
Cmd.ClearRecordFile

hasTMF(AID):
True if the targeted application contains a TMF.
False otherwise.

isCRIRequired(AID):
True if the targeted application requires CommitReaderID.
False otherwise.

CreateTMFCRI:
Cmd.Create TransactionMACFile requiring CRI

CreateTMFNoCRI:
Cmd.Create TransactionMACFile notrequiring CRI

SelectApplicationInvalid:
Cmd.SelectApplication (AID1, AID2)
AND has TMF (AID1) = True
AND has TMF (AID2) = True

SelectApplicationNoTMF1:
Cmd.SelectApplication(AID)
AND (has TMF (AID) = False
OR (TMC >= TMC Limit))

SelectApplicationNoTMF2:
Cmd.SelectApplication(AID1, AID2)
AND (has TMF (AID1) = False
OR (TMC >= TMC Limit))
AND (hasTMF(AID2) = False
OR (TMC >= TMC Limit))

SelectApplicationTMFCRI1:
Cmd.SelectApplication(AID)
AND has TMF (AID) = True
AND is CRIRequired (AID) = True

SelectApplicationTMFCRI2:
Cmd.SelectApplication(AID1, AID2)
AND has TMF (AID1) = True
AND has TMF (AID2) = False
AND is CRIRequired (AID1) = True

SelectApplicationTMFCRI3:
Cmd.SelectApplication(AID1, AID2)
AND has TMF (AID1) = False
AND has TMF (AID2) = True
AND is CRIRequired (AID2) = True

SelectApplicationTMFNoCRI1:
Cmd.SelectApplication(AID)
AND has TMF (AID) = True
AND isCRIRequired (AID) = False

SelectApplicationTMFNoCRI2:
Cmd.SelectApplication(AID1, AID2)
AND has TMF (AID1) = True
AND has TMF (AID2) = False
AND isCRIRequired (AID1) = False

SelectApplicationTMFNoCRI3:
Cmd.SelectApplication(AID1, AID2)
AND has TMF (AID1) = False
AND has TMF (AID2) = True
AND isCRIRequired (AID2) = False

TransactionCommit:
Cmd.CommitTransaction

TransactionAbort:
Cmd.AbortTransaction OR
Cmd.RollKeySet OR
Cmd.RestoreShadow

DeleteApplicationFile:
Cmd.DeleteApplication targeting selected application
OR Cmd.Format targeting TMAID
OR Cmd.DeleteFile targeting FileType. TransactionMAC

aaa-037629

Figure 18. Transaction MAC state diagram

Provided under IDA only - Personal copy for: COMPANY CONFIDENTIAL Sergi Salata Quantag ITS GmbH Do not copy, share or reproduce 5fc0779d-c2c-405 6ecc3ae4f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.14.3.4.2 Transaction MAC initiation

A Transaction MAC calculation is initiated on transaction start, if an application holding a FileType.TransactionMAC file is currently selected.

As outlined in Section 7.11.3, if multiple applications are selected, only one of them can hold a FileType.TransactionMAC file. Application selection will also fail in case the TMC inside the FileType.TransactionMAC file is currently 0xFFFFFFFF as no Transaction MAC computation is possible anymore.

Next to this, a new Transaction MAC calculation is initiated each time a transaction with ongoing Transaction MAC calculation is committed successfully with CommitTransaction, or aborted, as described in Section 7.14.2.

Initiating a Transaction MAC calculation consists of the following steps:

- Set TMI to the empty byte string.
- Set TMRICur to the empty byte string.
- Calculate the Transaction MAC Session Keys KeyID.SesTMMACKey and KeyID.SesTMENCKey, as defined in Section 7.14.3.2.3.

In case CommitReaderID is not required, maintaining the TMRI is not needed.

Note that AbortTransaction can be used to exclude data read (ReadData, ReadRecords or GetValue) from the Transaction MAC calculation if this data does not need to be authenticated via the Transaction MAC toward the backend.

7.14.3.4.3 Transaction MAC update

Once a Transaction MAC calculation is ongoing, the Transaction MAC Input TMI gets updated on each following data manipulation command targeting a file of any file type within the application containing the FileType.TransactionMAC, except FileType.TransactionMAC file itself.

The affected commands are listed below but the exact TMI updates are defined in the related command section.

- ReadData, see Section 7.13.1.1. Except when targeting the FileType.TransactionMAC file.
- WriteData, see Section 7.13.1.2.
- GetValue, see Section 7.13.2.1.
- Credit, see Section 7.13.2.2.
- LimitedCredit, see Section 7.13.2.4.
- Debit, see Section 7.13.2.3.
- ReadRecords, see Section 7.13.3.1.
- WriteRecord, see Section 7.13.3.2.
- UpdateRecord, see Section 7.13.3.3.
- ClearRecordFile, see Section 7.13.3.4

Note that for each of these commands always the plain data are added, independently from the actual communication settings and secure messaging (that is, plain data without any MAC, CRC, padding,. . .).

Individual files can be excluded from the TMI updates by enabling a TMI Exclusion FileMap via CreateTransactionMACFile or ChangeFileSettings. For an excluded file all read and write operations are excluded, i.e.all operations listed above.

The 4-byte *TMIExclFileMap* is interpreted as a bit map from which the LSB is sent first on the interface.

Individual bits are interpreted as follows:

- ’0’ indicating that the file is included in TMI updates. (default)
- ’1’ indicating that the file is excluded.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Bits are related as follows with the FileNo

- ⌊*FileNo*/8⌋ to find the byte number within the *TMIExclFileMap*
- *FileNo* mod 8 to find the bit within the byte, with 0 being the lsb

E.g. FileNo 9 is encoded in byte 1, bit 0.

Bits for which no file exists are ignored.

If multiple applications are selected, only the commands targeting a file in the application holding the FileType.TransactionMAC file are included in the calculation. Commands targeting files in the other application are ignored by the Transaction MAC computation. The *FileNo* is included as it appears on the command interface. This means that if the secondary application, as indicated by the SAI, is targeted, this is also included in the Transaction MAC Calculation.

The Transaction MAC feature does not affect the API of the data manipulation commands.

7.14.3.4.4 Commit ReaderID command

An application can be configured to require a reader to securely commit to a ReaderID during transaction. Typically this will require the support of a SAM inside the reader such that the ReaderID cannot be forged by a fraudulent attacker (and also the key required to commit the ReaderID cannot be known). Typically the ReaderID will be the SAM UID. The SAM command API will need to ensure that the ReaderID value itself cannot be manipulated by an attacker. In this case FileAR.ReadWrite of the FileType.TransactionMAC file will be configured to require authentication with a specific key.

As specified in FileType.TransactionMAC, FileAR.ReadWrite of the FileType.TransactionMAC file can also be configured to allow free CommitReaderID access. This allows for use cases where this command can be used to add reader data (ID, time, GPS location, etc.) to the Transaction MAC calculation. However, it should not be used for the use case where this command is used to be able to track unreported and missing transactions.

Committing a ReaderID is possible with the command CommitReaderID as defined in Section 8.9.3. Note that CommitTransaction will be rejected if CommitReaderID is required and has not been executed during the ongoing transaction. This is even the case if FileAR.ReadWrite of the FileType.TransactionMAC file is configured for free access, or if two applications are selected with one requiring CommitReaderID while all update operations were executed on the other application. Depending on the configuration of FileAR.ReadWrite of the FileType.TransactionMAC file, an authentication is required with with KeyID.AppCommitReaderIDKey.

If the selected applications do not hold such a file or committing the ReaderID is disabled according to this access right, the command is rejected.

The command is also rejected if a ReaderID has already been committed during the ongoing transaction.

The parameter TMRI is the committed ReaderID and is assigned internally to TMRICur. If authenticated, this value will be stored in non-volatile memory as TMRIPrev on a successful CommitTransaction. If not authenticated, the value is only used for the Transaction MAC computation.

The Transaction MAC Input TMI is updated as follows including the received ReaderID, that is, TMRICur, and if authenticated, the encrypted ReaderID of the latest successful transaction, that is, EncTMRI:

If authenticated: *TMI* = *TMI* || *Cmd* || *TMRICur* || *EncTMRI* || *ZeroPadding*

If not authenticated: *TMI* = *TMI* || *Cmd* || *TMRICur* || *ZeroPadding*

ZeroPadding is the minimal number of zero bytes added such that the length of the TMI up to and including the *ZeroPadding* is a multiple of the cipher block size used for the Transaction MAC calculation (that is, 0 until 15 bytes are added to get a multiple of 16 bytes). Note that this padding is also added if this TMI update is not the last one before CommitTransaction.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Original Data Solutions GmbH
Do not copy, share or reproduce
5fc07f90d-c22-4052-9016-cccc3ee42f1

If authenticated CommitReaderID will provide in its response the encrypted ReaderID of the previous transaction, as defined in Section 7.14.3.2.5. Note that this encryption is independent of the secure messaging as it is with a key derived from FileType.TransactionMAC. Once this ReaderID has been reported to the backend, the backend can decrypt and will be able to point to a specific merchant if the previous transaction has never been reported. If not authenticated, no data is returned and the given TMRICur will not be written to NVM. Note that even if FileAR.ReadWrite is configured for free access, it is possible to execute the command authenticated, which will trigger the TMRIPrev processing, that is, updating the value in NVM and replying the encrypted ReaderID.

The CommitReaderID requires CommMode.MAC if there is an active authentication. Information on authentication and secure messaging-dependent structure of the command can be found in Section 7.5.

7.14.3.4.5 Transaction MAC finalization

The Transaction MAC computation is successfully finalized on CommitTransaction as described in Section 7.14.2.1.

The Transaction MAC is computed as defined in Section 7.14.3.2.4.

If a transaction is aborted, either with AbortTransaction or implicitly by some other event, the ongoing Transaction MAC calculation is also aborted. This is described in Section 7.14.2.2

7.14.3.5 Transaction MAC and ISO/IEC 7816-4

The Transaction MAC feature can be used with ISO/IEC 7816-4 commands wrapping the native DESFire APDU format, as defined in Section 7.17.2. The behavior is the same as with the native commands. For example, TMI updates will exactly be as defined in Section 7.14.3.4.3 and the related command sections: only including the native command fields.

The Transaction MAC feature is not supported by the standard ISO/IEC 7816-4 commands. Also the ISO/IEC 7816-4 commands for EV Charging are not supported.

Therefore the following commands will return ISO6985 (Conditions of use not satisfied) when trying to select or access an application (DF) or a file within an application containing a FileType.TransactionMAC file.

- ISOSelectFile
- ISOReadBinary
- ISOUpdateBinary
- ISOReadRecord
- ISOAppendRecord
- VDE_ReadData
- VDE_WriteData

7.14.4 Transaction Signature

Similar as the Transaction MAC feature, also the Transaction Signature feature helps preventing fraudulent merchant attacks. It allows a merchant operating a point-of-sale terminal to prove that the transactions he executed with customers are genuine towards the card issuer's or application provider's backend. Being an asymmetriccrypto based feature, the public key can be shared more broadly, allowing potentially also the merchant (or its reader) to verify the Transaction Signature. As such the Transaction Signature can also be used to authenticate the card and transaction towards the reader executing the transaction.

In general, the Transaction Signature is very similar to the Transaction MAC feature, and reuses most of its concepts. Note that if a concept is used for both Transaction MAC and Signature, it may hold 'MAC' in its name, while applying also for Transaction Signature.

Provided under TIDA COMPANY CONFIDENTIAL - Personal copy for: certificata Solutions GmbH Do not copy, share or reproduce 5fc0770-922c-405c-90fe-6e23cae42f1

For Transaction Signature, the card will generate an ECDSA signature over the transaction with a KeyID.ECCPrivateKey of the application. This key is called the KeyID.AppTransactionSIGKey. As this key is not known to the merchant or its terminal, it is not possible for the merchant to forge signatures towards the backend.

The related public key can be shared with the card issuer's (or application provider's) backend. However, typically this public key will be authenticated through a certificate, signed by the backend's CA key pair. This certificate can be stored within the application during personalization and read-out during the transaction.

For replay protection, the Transaction MAC Counter concept of the the Transaction MAC feature is reused. To allow identifying non-reporting merchants, in the case of missing transactions, also the ReaderID committing feature from Transaction MAC is reused as is. Note that, also in the context of Transaction Signature, the reported ReaderIDs of the previous transaction are encrypted with symmetric crypto.

7.14.4.1 General overview

Similar to Transaction MAC, Transaction Signature is an application level feature that can be activated per application by creating FileType.TransactionMAC file. To enable Transaction Signature, one specifies the KeyID.AppTransactionSIGKey, see Section 7.12.4.6. This is a reference to a KeyID.ECCPrivateKey within the application. Similar as for Transaction MAC, one can optionally also require the commitment of a ReaderID, by assigning a KeyID.AppCommitReaderIDKey for this, see Section 7.12.2.

Once the Transaction Signature feature is activated, the Transaction Signature computation consists of the following steps:

1. **Initialization:** before or on the first command that is covered by the Transaction Signature (see below), the signature input is initialized with the empty string, see Section 7.14.4.2.
2. **Computation updates:** during the transaction, the Transaction Signature input is further extended with the data management commands and, depending on the configuration, also CommitReaderID. This is similar as for Transaction MAC, see Section 7.14.3.1. Note that for simplicity operations on all file types are included. Nevertheless, the feature does not really adds protection for FileType.StandardData, as those files can be written to without transaction finalization with CommitTransaction (see next step). Individual files can be excluded from the Transaction Signature calculation by configuring this through a filemap, either when creating the FileType.TransactionMAC file with CreateTransactionMACFile, or with ChangeFileSettings.
3. **Finalization:** on CommitTransaction, the signature input is finalized by adding the incremented Transaction MAC Counter (see Section 7.14.4.2.2), and the Transaction Signature is computed. The update of the manipulated backup files and the computed Transaction Signature, the related counter and (optionally) the committed ReaderID, are one atomic operation (as the files should not be updated without the Signature being available and vice versa). The computed Transaction Signature and the related counter are either provided with the response of CommitTransaction or can be read afterwards using ReadData on the FileType.TransactionMAC file.

7.14.4.2 Cryptographic and other concepts

7.14.4.2.1 Transaction MAC Counter

To mitigate replay attacks of the Transaction Signature, a counter equivalent to what is defined for Transaction MAC is used. For simplicity, the name is for both cases Transaction MAC Counter (TMC), as defined in Section 7.14.4.2.1.

In the Transaction Signature case, the incremented TMC is used as the final input for the Transaction Signature computation, see Section 7.14.4.4.2.

Provided under NXP CONFIDENTIAL Personal copy for: Sergi Sabata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc07791-c22c-4056-8ec3-cc3caedf1

7.14.4.2.2 Transaction MAC Counter Limit

Also in the case of Transaction Signature, the number of transactions that can be executed within an application can be limited by setting a Transaction MAC Counter Limit (TMCLimit). The behavior is equivalent as for Transaciton MAC, see Section 7.14.3.2.2.

7.14.4.2.3 Transaction Signature Session Keys

For the Transaction Signature, no session keys are used, as the private key of the configured KeyID.AppTransactionSIGKey is directly used for the signature computation. However, if the Transaction Signature is combined with an authenticated CommitReaderID, that is, if the FileAR.ReadWrite of the FileType.TransactionMAC is different from 0xE or 0xF, see Section 7.12.2, the KeyID.SesTMENCKey as defined in Section 7.14.4.2.3 is still used for the ReaderID encryption, see EncTMRI.

7.14.4.2.4 Transaction Signature Value and Input

The 64-byte Transaction Signature Value (TSV) is computed over the Transaction MAC Input (TMI) and the incremented Transaction MAC Counter (TMC+1). The Transaction MAC Input depends on the commands executed during the transaction, see CommitReaderID. The applied key is KeyID.AppTransactionSIGKey, that is, the KeyID.ECCPrivateKey associated with the FileType.TransactionMAC file. The applied algorithm is ECDSA Digital Signature Generation as defined in [28]. The hash function to be applied is SHA-256, as specified in NIST FIPS 180-4 [25].

The TSV is calculated as follows:

$$TSV = ECDSA_{Sign}(K_{AppTransactionSIG}, TMI \,||\, TMC + 1))$$

7.14.4.2.5 Transaction Signature ReaderID and its encryption

In case CommitReaderID is enabled, the ReaderID processing and encryption is equivalent as in the Transaction MAC case, see Section 7.14.3.2.5.

Note that this means that for this encryption symmetric crypto is used. This means a KeyID.AppTransactionMACKey must have been provided on FileType.TransactionMAC creation, from which the derived KeyID.SesTMENCKey is applied.

7.14.4.3 Transaction Signature enabling

The Transaction Signature feature is enabled for an application if the application holds an FileType.TransactionMAC creation file configured for Transaction Signature.

This is similar as for Transaction MAC, see Section 7.14.3.3.

7.14.4.4 Transaction Signature calculation

7.14.4.4.1 Transaction Signature state diagram

The Transaction Signature state diagram is equivalent to the Transaction MAC state diagram specified in Section 7.14.3.4.1, except for the actions ActInitTM and ActFinalTM which are redefined in the subsequent sections.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.14.4.4.2 Transaction Signature initiation

On application selection, it is checked if the application (or one of the applications) holds a TMI file configured for Transaction Signature. If this is the case, but the KeyID.AppTransactionSIGKey (that is, the referenced KeyID.ECCPrivateKey) does not exist, is not configured for Transaction Signature, or has an enabled KeyUsageCtrLimit that was already reached, see Section 7.9.1.2, the application is only selected with limited functionality, similar as when the TMCLimit is reached (see Section 7.14.3.2.2). This means the application can still be selected, but ISOSelectFile will be responded with ISO6283 and SelectApplication will return OPERATION_OK_LIM. CommitTransaction will be rejected, that is, no further transactions can be executed.

If the TMI file is correctly configured for Transaction Signature is currently selected the transaction will be initiated. This is shown in Figure 18 by the arrows triggering the action ActInitTM. Also, a new Transaction Signature calculation is initiated each time a transaction with ongoing Transaction MAC calculation is committed successfully with CommitTransaction, or aborted, as described in Section 7.14.2.

Initiating a Transaction Signature calculation consists of the following steps:

- Set TMI to the empty byte string.
- Set TMRICur to the empty byte string.

If authenticated CommitReaderID is enabled, an implementation may also already compute the KeyID.SesTMENCKey, though this can also be delayed till the CommitReaderID command itself.

7.14.4.4.3 Transaction Signature update

Once a Transaction Signature calculation is ongoing, the Transaction MAC Input TMI gets updated on each following data manipulation command targeting a file of any file type within the application containing the FileType.TransactionMAC, except FileType.TransactionMAC file itself.

Individual files can be excluded from the TMI updates by enabling a TMI Exclusion FileMap via FileType.TransactionMAC or ChangeFileSettings.

This is equivalent as for Transaction MAC, see Section 7.14.3.4.3.

7.14.4.4.4 CommitReaderID

The CommitReaderID is equivalent as for Transaction MAC, see Section 7.14.3.4.4.

7.14.4.4.5 Transaction Signature finalization

The Transaction Signature computation is successfully finalized on CommitTransaction as described in Section 7.14.2.1. This is shown in Figure 18 by the arrows triggering the action ActFinalTM.

The TMI is completed by concatenating the incremented TMC, that is, a 4-byte string holding the following value: (TMC + 1).

The Transaction Signature is computed as defined in Section 7.14.4.2.4.

If a transaction is aborted, either with AbortTransaction or implicitly by some other event, the ongoing Transaction Signature calculation is also aborted. This is described in Section 7.14.2.2.

7.14.4.5 Transaction Signature and ISO/IEC 7816-4

Similar as Transaction MAC, the Transaction Signature feature can be used with ISO/IEC 7816-4 commands wrapping the native DESFire APDU format, but not with standard ISO/IEC 7816-4 commands. See Section 7.14.3.5.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.14.4.6 Authentication vs Transaction MAC/Signature

MF3E(H)x3 has no limitations on what type of authentication can be combined with either Transaction MAC or Transaction Signature. This allows to fulfill different use cases depending on the cryptographic algorithms that are supported in the reader or the backend. This means that any of the following combinations is supported:

- Asymmetric mutual authentication initiated by ISOGeneralAuthenticate or ISOSelectFile in an application with a FileType.TransactionMAC enabled for Transaction Signature.
- Asymmetric mutual authentication initiated by ISOGeneralAuthenticate or ISOSelectFile in an application with a FileType.TransactionMAC enabled for Transaction MAC.
- Asymmetric reader-unilateral authentication initiated by ISOGeneralAuthenticate or ISOSelectFile in an application with a FileType.TransactionMAC enabled for Transaction Signature. Note that may offer an efficient use of the required ECC operations. However, one should take special care as the card is not yet authenticated in the secure channel. Therefore a reader should not make security critical decisions based on the data read and write responses until the Transaction Signature has been received and validated.
- Asymmetric reader-unilateral authentication initiated by ISOGeneralAuthenticate or ISOSelectFile in an application with a FileType.TransactionMAC enabled for Transaction MAC.
- Symmetric mutual authentication initiated by AuthenticateEV2First and AuthenticateEV2First in an application with a FileType.TransactionMAC enabled for Transaction Signature.
- Symmetric mutual authentication initiated by AuthenticateEV2First and AuthenticateEV2First in an application with a FileType.TransactionMAC enabled for Transaction MAC.
- Transactions without any authentication in an application with a FileType.TransactionMAC enabled for Transaction Signature. This could be a free CommitReaderID, allowing a very short transaction where the card just authenticates some data received from the reader.
- Transactions without any authentication in an application with a FileType.TransactionMAC enabled for Transaction MAC.

7.14.5 Transaction Timer

To mitigate attacks where a Man-in-the-Middle (MitM) attacker would delay execution of CommitTransaction or a final write command to complete a transaction on the card, by keeping the card powered until, for example, they are being controlled by a control agent during public transport, MF3E(H)x3 supports a transaction timer feature. This feature allows the card issuer to configure a maximum time a transaction can take. Once the threshold is exceeded, the card will automatically reset.

The transaction timer shall be configured by setting TransactionTimer with SetConfiguration Option 0x05 via the PDCap2.2 (see Section 7.7.3.1) to a value different from 0x00. Three different values are defined:

- 0x01: timer will expire between 1 second and 3 seconds.
- 0x02: timer will expire between 10 seconds and 30 seconds.
- 0x03: timer will expire between 100 seconds and 300 seconds.

In normal processing, the expiration should be close to the lower boundary. Upper boundaries are relevant in case of an attacker playing specific tricks like injecting erroneous I-frames, etc.

Once enabled, the timer shall be activated on any subsequent application selection with SelectApplication or ISOSelectFile. If two applications are selected, the most strict configuration is taken, that is, if both application enable the feature the threshold with the lowest value is used. In case of ISOSelectFile for virtual card selection and AuthVCMandatory and/or PCMandatory is set, the timer is activated on the first command after the VC selection and/or Proximity Check. In case of implicit application selection of the application with native AID=0xEEE000, the timer shall be started on a first ISO/IEC 7916-4 APDU different from ISOSelectFile. The feature cannot be enabled for the PICC level.

Provided under NDA by NXP Semiconductors GmbH
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Stalla
Do not copy, share or reproduce
Quality IT Solutions GmbH
5fc07f9d-62c-402c-90fe-6eccc8ae4221

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

If the timer expires, the card will be reset similar as with full power-off reset. Over NFC, this means moving the card into ISO/IEC 14443-3 IDLE state. Over I^2C, this means selecting the PICC level, aborting the application authentication and transaction.

As long as the timer has not expired, regular command execution is not influenced by the feature.

Once enabled, the transaction timer value shall be reflected via the PDCap2.2 as returned on a AuthenticateEV2First, see Section 7.20.4.1.1.

If reflection on AuthenticateEV2First is not sufficient, one can consider encoding it in the FCI as well via user configuration.

Note that one needs to take care that timer expiration does not occur during normal transaction execution, for example, for use cases where a card resides on a reader for longer time. If required the PCD can actively trigger a timer reset via power-off reset, ISO/IEC 14443-4 Deselect and re-activation or a new application selection.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.15 Crypto API

MF3E(H)x3 supports generic execution of cryptographic primitives via the CryptoRequest command. The following operations are supported:

- CryptoRequest_ECCSign: ECC signature generation with ECDSA, see Section 7.15.1.
- CryptoRequest_Echo: Echo operation, see Section 7.15.2.

MF3E(H)x3 only supports a single cryptographic operation, but the API is defined to allow further extensions. CryptoRequest, as defined in Section 8.10.1 and Figure 80 specifies the general command structure and generic error conditions, while the subcommand definitions specify the actual operations.

CryptoRequest requires the access condition and communication mode as defined in the Crypto API management (see SetConfiguration Option 0x15). By default, CommMode.MAC is applied, requiring authentication granting KeyID.AppMasterKey, or at PICC-level KeyID.PICCMasterKey access rights.

7.15.1 ECCSign

The command CryptoRequest_ECCSign as defined in Section 8.10.2 and Figure 81 supports signing a precomputed hash, or raw data, which means the hash operation is executed by MF3E(H)x3 on the fly. If the data is raw, the signature can be computed via a one-shot operation, or in the case of bigger input data, via an init-update-final mechanism. This is encoded via Operation.

As set by Algorithm, the command implements the ECDSA Digital Signature Generation as defined in [28]. The supported hash function is SHA-256, as specified in NIST FIPS 180-4 [25].

The key targeted by KeyNo is used to compute the signature. If the targeted KeyID.ECCPrivateKey does not exist or is not enabled for ECC Sign, the command is rejected. If the key has an enabled KeyUsageCtrLimit that was already reached, see Section 7.9.1.2, the command is also rejected.

InputSource is reserved for future use.

InputLen encodes the input length. The maximum supported input length for a frame is limited to ensure the data still fits in ISO/IEC 7816-4 APDU if wrapped. No native or ISO/IEC 14443-4 chaining is supported for bigger sizes as the init-update-final approach shall be used. If Operation is configured for hash input, InputLen shall be 0x20. InputData shall hold the actual raw input data or hash.

7.15.2 Echo

The command CryptoRequest_Echo as defined in Section 8.10.3 and Figure 82 supports returning provided command data. This allows to easily test the communication interface.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-445c-9008-6ecc3caa2f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.16 GPIO management

In specific product types / product configurations of MF3E(H)x3, GPIOs are available and can be used.

Please refer to Section 2.2, for the availablility of the GPIOs in different package types.

If the GPIOs are available in the product configuration, following points apply:

MF3E(H)x3 supports two GPIOs:

- GPIO1 may be configured for input detection of a binary-state input signal (for example, to detect if a button is pressed or not), binary-state output signal or down-stream power-out.
- GPIO2 may be configured as a binary-state input or output signal (with or without NFC Pause file, see below).

GPIO configuration is done with SetConfiguration 0x11, see Section 7.7.3.1, and especially Table 25. For each of the modes, dedicated HW aspects as outlined in Table 25 can be set. When configured for down-stream power-out, a targeted voltage/current level needs to be set. It is however possible to overwrite this at runtime via ManageGPIO, if, for example, not sufficient power can be harvested from the actual field strength. When configured as output, the GPIO can also be configured to notify on authentication. This is further detailed in Section 7.16.5.

The ReadGPIO command, as defined in Section 7.16.3, may be used to read the current status of the GPIOs. The ManageGPIO command, as defined in Section 7.16.2, is used for controlling the output on GPIO1. In addition to supporting external device notification, the ManageGPIO may be used to retrieve information from an external MCU via the 'NFC Pause' option, see Section 7.16.1.

7.16.1 NFC Pause feature

The NFC Pause feature allows to transfer control from an NFC Host to an MCU controlling MF3E(H)x3 as a controller via the I^2C interface.

Two ways to activate an NFC Pause are supported:

- ManageGPIO can trigger the NFC Pause via its parameters. This option is possible for any GPIO that is configured for output, including GPIO2 if configured with NFCPauseFile, see SetConfiguration 0x11. This flow is outlined in Figure 19.In this case, the GPIO output signal is also explicitly controlled through the issued ManageGPIO.
- ISOReadBinary/ReadData can implicitly trigger the NFC Pause when targeting the file that is configuredfor NFC Pause. This allows to use the feature in the context of an NFC Forum Type4Tag [10], as outlined in Figure 20. This option is only available for GPIO2, as explicitly configured with SetConfiguration 0x11. In this case, the GPIO2 output is automatically toggled. MF3E(H)x3 supports configuring a single file , within a specified application, together with an start and end offset. The NFC Pause will only be triggered if any data withing those offsets is read, that is, *NFCPauseOffset*<(*ReadOffset*+*ReadLength*) and *NFCPauseOffset*+ *NFCPauseLength* > *ReadOffset*. In a Type4Tag context, this allows for example, triggering the NFC Pause only when reading NDEF length (to allow dynamic NDEF length being returned) or only when reading the actual content, avoiding multiple switches to the MCU.
  Note that in case of multiple application selection, the NFCPause through read commands can only be triggered on the primary application. On the secondary application, the data of the file is returned without triggering an NFC Pause.
  In case of chaining of the read response (be it native or ISO/IEC 14443-4 chaining), the NFC Pause is only triggered on the first frame.

When NFC Pause is triggered, MF3E(H)x3 shall halt processing on the NFC interface (only sending WTx requests) until the ManageGPIO is received on the I^2C interface to indicate re-start of NFC processing, that is, releasing the NFC Pause.

Until the NFC Pause Release, the MCU can send any command over the I^2C interface. When NFC Pause is triggered by ISOReadBinary/ReadData, this will typically include updating the targeted file content through

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Sadovnychyi
Quantagonia Solutions GmbH
Do not copy, share or reproduce
5f07779d-022b-4b7e-9b6e-6ecc3c4c42f1

ISOUpdateBinary/WriteData. As the GPIO only provides a binary signal, the MCU does not know what command triggered the NFC Pause, and in case of a read command what offset and length are requested.

When NFC Pause is released, MF3E(H)x3 behavior depends on how the NFC Pause was triggered. If triggered by ManageGPIO, MF3E(H)x3 shall use the NFCPauseRespData provided over the I^2C interface in the release command as the response data provided on the NFC interface. The size of this data is limited up to 239 byte. If the NFC Pause was triggered by ISOReadBinary/ReadData, MF3E(H)x3 will ignore the NFCPauseRespData (if any), and retrieve the NFC response data by further processing the received read command (that is, from the targeted file).

On NFC Pause, the application selection and transaction state is transferred from the NFC interface to the I^2C interface.Therefore there is no need to re-select the targeted application over I^2C. Potential uncommitted write operations from the NFC interface can be committed over I^2C. The authentication state is not transferred, that is, over I^2C one starts in VCState.NotAuthenticated. If a new authentication is initiated over I^2C, any active authentication on the NFC interface is lost.

On NFC Pause Release, the same is true: application selection and transaction state is transferred from the I^2C interface to the NFC interface. The original authentication state from before the NFC pause is still active over the NFC interface, except if a new authentication was initiated over I^2C. Then the NFC interface also starts again in VCState.NotAuthenticated.

This means MF3E(H)x3 can only maintain a single authentication session, giving the following options:

- if triggered by ISOReadBinary: as there is then no authentication over the NFC interface, authentication over I^2C is possible, and may be required for for example, executing WriteData.
- if triggered by ReadData: depending on the targeted file's access conditions, an authentication over NFC interface may be required. If required, one must not require an authentication for WriteData over I^2C, that is, FileAR.Write or FileAR.ReadWrite shall typically be set to 0xD allowing free access over I^2C. If an authentication is initiated over I^2C, the NFC authentication shall be lost. Therefore, after transferring the control back to NFC, the ReadData may fail, returning an error, as access control checks are done again this stage. Note that when the communication mode changed during the NFC Pause this updated mode is immediately applied on the response.
- if triggered by ManageGPIO: depending on the ManageGPIOAccessCondition, an authentication may be required. If authenticated over the NFC interface, and an authentication is initiated over I^2C, the NFC authenticationshall be lost. Similar as above, after transferring the control back to NFC, the ManageGPIO may fail, returning an error, as access control checks are done again this stage.

For ISOReadBinary and ReadData, if the application state was changed over I^2C, the read command will still be rejected over NFC as the targeted file does not reside in the currently selected applicaiton anymore. ISOReadBinary/ReadData when triggering NFC Pause over the NFC interface, will not trigger special processing when targeting the same data over I^2C.

Note that Figure 19 is only one possible example. More complex flows are also possible. For example, one could set the GPIO to HIGH on a first ManageGPIO without pausing the NFC. This could trigger the Controller MCU to start its processing (for example, interrogating an I^2C target sensor), while in parallel on the NFC interface with MF3E(H)x3, the NFC host can still further exchange commands to read or updates some files. Once the NFC host is done, it can send another ManageGPIO setting the GPIO back to LOW while pausing the NFC. This then triggers the Controller MCU that it can return its data over the NFC interface.

Provided under NDA to: CONFIDENTIAL Personal copy for: Sergii Gatsa Quantas Solutions GmbH Do not copy, share or reproduce 5fc0779d-2220-465c-90fe-6ec3cae4251

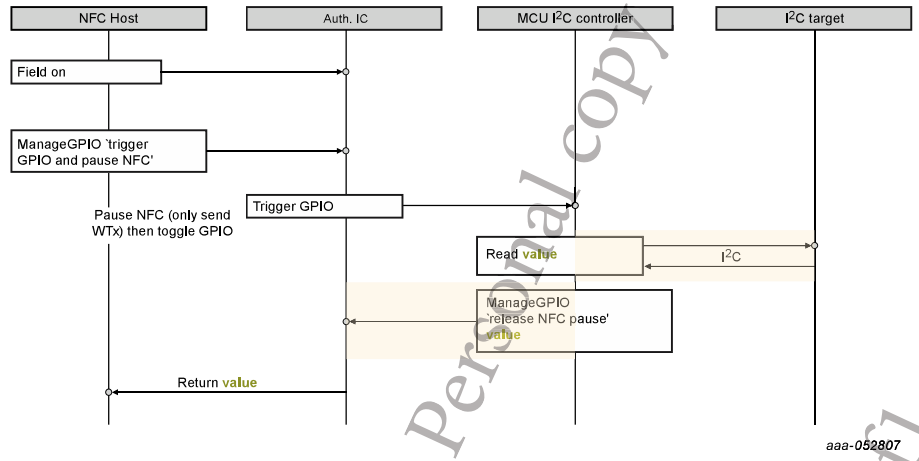


Figure 19. NFC Pause Example

The Figure 20 gives a further example where the NFC Pause is triggered by an ISOReadBinary. Note that the figure still shows a simplified flow, as a Type4Tag is typically read in multiple stages (first reading the CC file and NDEF length) [10].

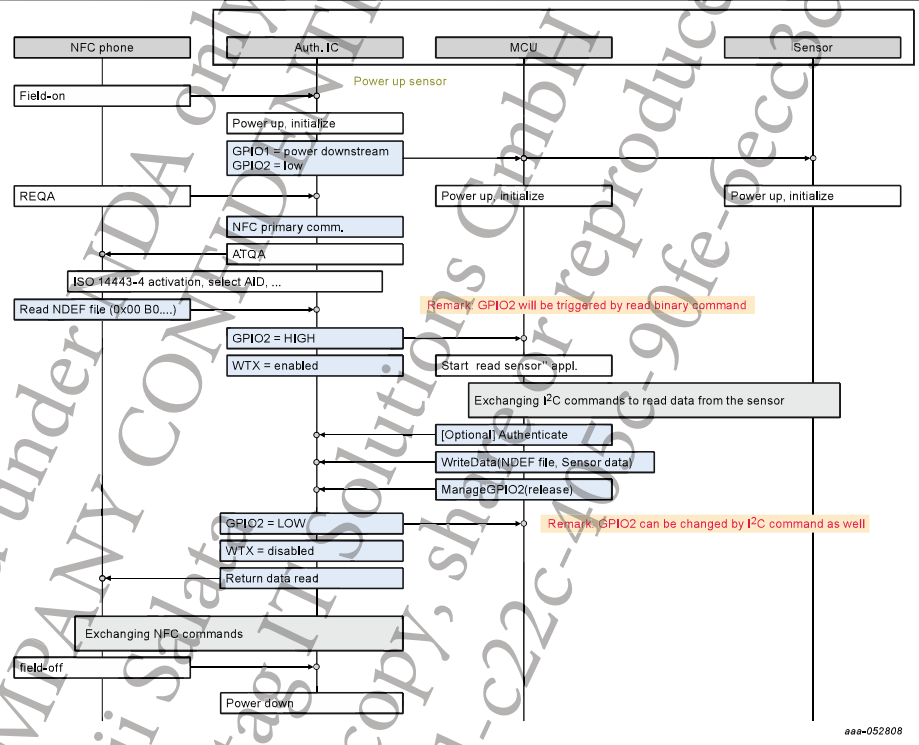


Figure 20. NFC Pause Example with ISOReadBinary

For combining SDMMAC/SDMSIG with the NFC Pause feature, to ensure a consistent MAC or Signature, the SDM file must be carefully configured. Either the file should be read in one shot, or the file shall be configured and read such that the NFC Pause is only triggered in the first read. For example, following the [10] read procedure, the NFC Pause could be triggered on the first read command if the file is properly configured via setting NFCPauseOffset and NFCPauseLength to the NDEF length segment. At the same time, the SDMMAC/SDMSIG should only be influenced by subsequent read commands, i.e. after the NFC Pause,

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Cesgii Salate
Do not copy, share or reproduce
DigitalTrust Solutions GmbH
5f07779d-c22c-4056-90fe-6ecc3ca4211

when reading the actual NDEF message, by configuring the SDMMACInputOffset after the NDEF length segment. If not, as the NFC Pause may then be interrupting the sequence of read commands, dynamic data or re-generation of the ECDSA ephemeral key could lead to inconsistent SDMMAC/SDMSIG.

7.16.2 Manage GPIO

The ManageGPIO command format is outlined in Section 8.11.1 and Figure 83. The command is accepted at both application level and PICC level (that is, AID 0x000000).

Depending on the ManageGPIOAccessCondition, for PICC level, and at application level addAID and addManageGPIOAccessCondition, as configured with SetConfiguration Option 0x11, the command may require an authentication and the configured secure messaging communication mode. By default, the command is disabled. The Release NFC Pause action shall only be accepted over the I^2C interface and does not require authentication independently of the configured access condition.

When a GPIO is configured for output, after a power-on reset, the GPIO will be initialized with the state as configured by the GPIOXConfig from SetConfiguration Option 0x11 on the first command after activation, that is, ISO/IEC14443-4 or I^2C activation. This is independently of the state from a previous activation. Immediately after the power-on reset, output GPIOs will be in high-impedance (High-Z) state.

Similarly, when GPIO1 is configured for down-stream power out, after a power-on reset, the feature will always be disabled by having the pin in high impedance (High-Z) state, independently of the state from a previous activation. By default, when enabling down-stream power out via the SET operation, the voltage/current level as configured with SetConfiguration Option 0x11 will be targeted. It is also possible to give a different target at runtime via ManageGPIO. This allows reacting on a WEAK_FIELD response. If only a targeted voltage level is given, the measured current will also be returned. Alternatively, as long as power downstream was not enabled, one could execute a MEASURE operation. If combined with CLEAR, power downstream is kept disabled, that is, only a measurement is done. If combined with SET, this enables down-stream power out at the same time, for example, to already start power downstream with a conservative target. Potentially, one can then later increase the targeted power by a further ManageGPIO SET operation based on the received measurement. Note that once down-stream power out has been enabled, a further MEASURE operation will be rejected. If enabling power downstream, the targeted power will be limited according to the in rush current limitation and duration configured with SetConfiguration Option 0x11. It is also possible to reserve part of the available RF Power for MF3E(H)x3 via the AdditionalCurrent configuration. When executing a ManageGPIO SET or MEASURE operation, or the combination of both, MF3E(H)x3 always returns a WTX over the NFC interface, before executing the operation. Additionally, the command allows to configure the automated WTX sending at run-time, e.g. for a subsequent NFC Pause operation. To enable automated WTX sending, the MF3E(H)x3 must be VCC powered or power downstream must be active.

7.16.3 Read GPIO

The ReadGPIO, as defined in Section 8.11.2 and Figure 84, returns the status of GPIO1 and/or GPIO2 for both input and output use cases, as configured with SetConfiguration Option 0x11, see Section 7.7.3.1.

For GPIO input configurations (GPIOXMode = 0x01), only the current status for GPIO1 and GPIO2 is returned: GPIO1CurrStatus and GPIO2CurrStatus. This value is the value as measured during the execution of the ReadGPIO command.

For GPIO output (GPIOXMode = 0x02 or GPIO2Mode = 0x05) and down-stream power out (GPIO1Mode = 0x04) operations, the current status can be retrieved. That is, whether the output or down-stream power out has been set or not. This allows an external host to keep track.

For both input and output cases, GPIO1CurrStatus and GPIO2CurrStatus can take the following values:

- *Low*: 0x4C, that is, the ASCII encoding of ‘L’. This is the value for a logical ‘0’, for example, if the button is not pressed. In case of output, the output is not driven, or down-stream power out is not enabled (for example, after CLEAR operation with ManageGPIO).

Provided under NDA to COMPANY CONFIDENTIAL Sergii Gatta Quantum Software GmbH Do not copy, share or distribute 5fc07790-0ZC-4052-9006-6ec0ca3a4A1 Personal copy for:

- *High*: 0x48 , that is, ASCII encoding of ‘H’. This is the value for a logical ‘1’, for example, if the button is pressed. In case of output, the output is driven or down-stream power out is enabled (for example, after SET operation with ManageGPIO).

Note that all output cases for a GPIO pin configuration are covered in a single row in the table below. The following value is returned if the GPIO pin is disabled, or the feature is not enabled yet:

- *Invalid*: 0x49, that is, the ASCII encoding of ‘I’. This is the value when the feature has not been enabled. The complete GPIO status is returned on three bytes:
- Byte[0]: Reserved (that is, N/A and therefore always ‘I’)
- Byte[1]: GPIO1CurrStatus or N/A
- Byte[2]: GPIO2CurrStatus or N/A

This results in the following possible outputs, depending on the GPIO configuration:

Table 47. Read GPIO response

Configuration		Responsedata		
GPIO1Conf	**GPIO2Conf** Input	**GPIOByte0** Reserved =(‘I’)	**GPIOByte1** GPIO1CurrStatus =(‘H’/‘L’)	**GPIOByte2** GPIO2CurrStatus =(‘H’/‘L’)
Input	Output	Reserved =(‘I’)	GPIO1CurrStatus =(‘H’/‘L’)	GPIO2CurrStatus =(‘H’/‘L’)
	Other	Reserved =(‘I’)	GPIO1CurrStatus =(‘H’/‘L’)	N/A =(‘I’)
Output	Input Output Other	Reserved =(‘I’)	GPIO1CurrStatus =(‘H’/‘L’)	GPIO2CurrStatus =(‘H’/‘L’)
		Reserved =(‘I’)	GPIO1CurrStatus =(‘H’/‘L’)	GPIO2CurrStatus =(‘H’/‘L’)
		Reserved =(‘I’)	GPIO1CurrStatus =(‘H’/‘L’)	N/A =(‘I’)
Other	Input Output Other	Reserved =(‘I’)	N/A =(‘I’)	GPIO2CurrStatus =(‘H’/‘L’)
		Reserved =(‘I’)	N/A =(‘I’)	GPIO2CurrStatus =(‘H’/‘L’)
		Reserved =(‘I’)	N/A =(‘I’)	N/A =(‘I’)

The command is accepted at both application level and PICC level (that is, AID 0x000000).

Depending on the ReadGPIOAccessCondition, for PICC level, and at application level addAID and addReadGPIOAccessCondition, as configured with SetConfiguration Option 0x11, the command may require an authentication and specific secure messaging communication mode. By default, the command is disabled.

7.16.4 Mirroring in the NDEF message

The GPIO Input statuses can be mirrored together within the NDEF messaging. In this way, the status can be protected by the Secure Dynamic Messaging, as defined in Section 7.5.11 and more specifically Section 7.5.11.1.5.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Vilda
Quantum Solutions GmbH
Do not copy, share or reproduce
5fc0779d-022c-4055-90fe-6ecc3c2a2f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

If mirroring is enabled, the encoding within the NDEF file will be identical to the 3-byte ReadGPIO response, see Table 47. Note that only Input configurations are mirrored. Output configurations are interpreted as 'Other' for NDEF mirroring, that is, returning 'I' for Invalid.

To enable this mirroring, the configuration needs to be done with ChangeFileSettings, see Section 7.12.2.3.

7.16.5 Authentication notification

When configured as output, the GPIO can be configured to notify on authentication. This configuration is also done via SetConfiguration 0x11 using the *GPIO1Notif* and *GPIO2Notif* parameter.

When configured, the targeted GPIO will be enabled (that is, set to HIGH representing logical '1') once the authenticated state is reached. This means:

- On successful execution of ECC-based mutual or reader-unilateral authentication with ISOGeneralAuthenticatePart2 or ISOGeneralAuthenticateFinal.
- On successful execution of symmetric mutual authentication with AuthenticateEV2FirstPart2 or AuthenticateEV2NonFirstPart2.

Note that it is still possible to manually toggle the GPIO with ManageGPIO (potentially even triggering an NFC Pause) if authentication notification is enabled. Most likely, this kind of double usage of a GPIO pin should be avoided for a use case, as can for example, be done by configuring the ManageGPIOAccessCondition to 0xF.

When losing the authentication state, the GPIO will be disabled (i.e.set to LOW representing logical '0'). See Section 7.5.3 and Section 7.5.4 for the different reasons to lose authentication.

7.16.6 NFC field notification

When configured as output, the GPIO can be configured to notify on NFC filed presence. This configuration is also done via SetConfiguration 0x11 using the *GPIO1Notif* and *GPIO2Notif* parameter.

When configured, the targeted GPIO will be enabled (that is, set to HIGH representing logical '1') once the MF3E(H)x3 detect an NFC field.

When MF3E(H)x3 is removed from the NFC field, this has no effect on the GPIO notification. It is assumed that an external MCU acts on the enabling of the GPIO pin by either halting the ongoing I^2C session. In this case, a reset cycle would clear the GPIO pin, reenabling communication interface arbitration. In case the MCU ignores the GPIO2 signal, it must be understood that it will remain set until, for example, manually cleared with ManageGPIO.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.17 ISO/IEC 7816-4 Support

Next to the native command set, MF3E(H)x3 also supports ISO/IEC 7816-4 commands [10]. MF3E(H)x3 supports an optional wrapping into ISO/IEC 7816-4 APDUs of the native command set (including Proximity Checking, see Section 7.18). On top, the following standard ISO/IEC 7816-4 commands are supported as well:

- ISOSelectFile with INS code 0xA4
- ISOReadBinary with INS code 0xB0
- ISOUpdateBinary with INS code 0xD6
- ISOReadRecord with INS code 0xB2
- ISOAppendRecord with INS code 0xE2
- ISOGetChallenge with INS code 0x84

7.17.1 ISO/IEC 7816-4 command format

MF3E(H)x3 supports the APDU message structure according to ISO/IEC 7816-4 [10] for

- an optional wrapping of the native APDU format, see Section 7.17.2.
- for the additionally implemented standard 7816-4 commands, see Section 7.17.3.

MF3E(H)x3 uses for all ISO/IEC 7816-4 commands the format as given in Figure 21.

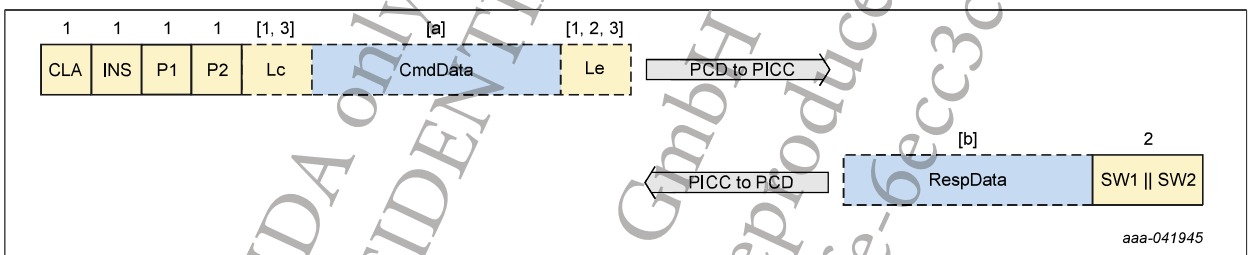


Figure 21. ISO/IEC 7816-4 command format

Parameters *CLA*, *INS*, *P1* and *P2* are detailed in every command description.

MF3E(H)x3 does not support logical channels as indicated in *CLA*. *CLA* will also have no indication on whether secure messaging is applied.

Lc states the length of the data parameter. It is only present if there exists a data in the command stream.

Le states the length of the data that should be sent back from the card. If *Le* == 0x00, then all data should be sent back, which is available. Exception is wrapping, see Section 7.17.2, where only 0x00 can be used.

MF3E(H)x3 supports extended length fields according to [10].

ISO/IEC 7816-4 commands can make use of the larger FSC as advertised by the ATS, see Command Size.

This holds for both standard ISO/IEC 7816-4 commands and wrapped commands (in case the wrapped native command supports the full FSC).

MF3E(H)x3 supports both ISO/IEC 14443-4 chaining as defined in Section 7.5.2.1 and native chaining, that is, for wrapped commands, as defined in Section 7.5.2.2.

For more information on the principle structure of APDUs in ISO/IEC 7816-4, see [10].

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.17.2 Wrapping of Native APDUs

Wrapping of the native MF3E(H)x3 APDU Format is done as given in Figure 22.

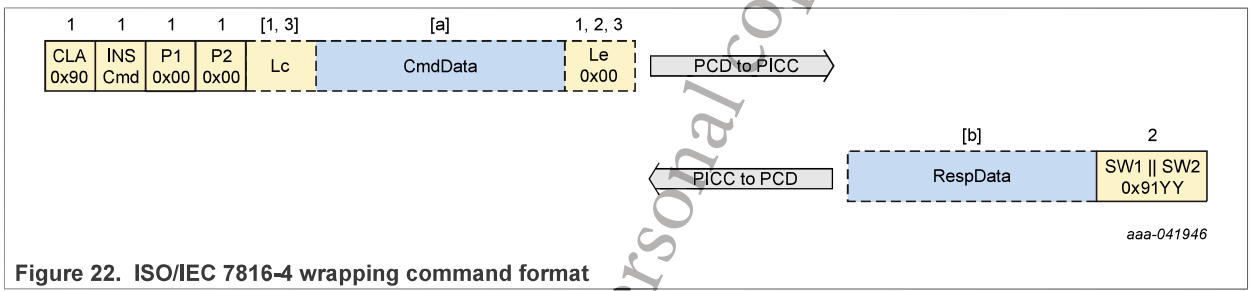


Figure 22. ISO/IEC 7816-4 wrapping command format

CLA shall always be set to 0x90.

INS shall be set to the native command code (*Cmd*).

P1 and *P2* shall always be set to 0x00.

Lc shall be set to the length of the wrapped data, if present.

CmdData shall be set to the remaining native command parameters, if any. Note thus that he ISO/IEC 7816-4 CmdData consists of native *CmdHeader* and *CmdData* if present.

Le shall be set to 0x00, which defines that any length of the PICC response is allowed. Other values than 0x00 are not allowed. The *Le* byte has to be always present.

The length of the total unwrapped native command is limited to 55 bytes, except for the data management commands that can make use of larger FSC:

- ReadData with Cmd 0xAD
- WriteData with Cmd 0x8D
- ReadRecords with Cmd 0xAB
- WriteRecord with Cmd 0x8B
- UpdateRecord with Cmd 0xBA
- ChangeFileSettings with Cmd 0x5F

For these commands the only limitation for single frames is the advertised FSC size, see also Command Size.

For the response, the native *RespData* is concatenated with *SW1*||*SW2*. *SW1* shall be set to 0x91, while *SW2* will hold the native response code. So in case of successful execution, OPERATION_OK is represented by 0x9100 and ADDITIONAL_FRAME by 0x91AF.

Secure messaging is applied exactly in the same way as described for the native command format, i.e. using the ISO/IEC 7816-4 *INS* and *CmdData* (mapping the native Cmd, *CmdHeader* and *CmdData*) for commands, and the ISO/IEC7816-4 SW2 and *RespData* (mapping the native RC and *RespData*) for responses respectively.

Next to the (wrapped) native error code, see Section 7.5.3 and the individual command descriptions, the following generic ISO/IEC 7816-4 APDU-related errors can occur:

- Wrong or inconsistent APDU length according to [10]. In this case the wrapped LENGTH_ERROR will be returned: 0x917E.
- More data requested than allowed by Le. In this case the wrapped LENGTH_ERROR will be returned: 0x917E. An example is when trying to read from FileType.StandardData file while the response does not fit in 256 bytes while short Le = 0x00 is sent. Note that secure messaging overhead might add on top of the requested file content.
- Incorrect parameters P1-P2. ISO6A86 will be returned.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

In case of unsuccessful command execution, MF3E(H)x3 will execute the same abort actions as for native commands, see Section 7.5.3.

7.17.3 Standard ISO/IEC 7816-4 commands

MF3E(H)x3 supports a selection of standard ISO/IEC 7816-4 commands, that is, commands from the interindustry class [10]. These commands are defined in this section. First the authentication and secure messaging aspects are of these commands are described.

7.17.3.1 Byte order

For all parameters of standard ISO/IEC 7816-4 commands, the representation on the interface is most significant byte (MSB) first notation. As data like the 2-byte ISO/IEC 7816-4 file identifiers, are in different order during file and application creation with native commands, this needs to be especially taken into account.

7.17.3.2 Security Concept of ISO/IEC 7816-4 commands

MF3E(H)x3 supports a mutual authentication with standard ISO/IEC 7816-4 commands, as specified in Section 7.5.6. Successful authentication moves the PICC into the VCState.AuthenticatedECC, see Section 7.5.5.

Once authenticated, it is possible to issue wrapped native commands using the complete EV2 secure messaging applied on the unwrapped command, as defined in Section 7.5.10.

Data operations executed with standard ISO/IEC 7816-4 commands do not support secure messaging and will always be exchanged in plain. Targeted use cases are personalization in secure environment, NTAG Type 4 support, and other use cases that do not require authentication and secure messaging. These data management commands are only supported in VCState.NotAuthenticated. In VCState.AuthenticatedAES and VCState.AuthenticatedECC, the following standard ISO/IEC 7816-4 commands are thus rejected:

- ISOReadBinary
- ISOUpdateBinary
- ISOReadRecord
- ISOAppendRecord

7.17.3.3 Error handling

In case of unsuccessful command execution, MF3E(H)x3 will send a return code different from ISO9000>. The full list of ISO/IEC 7816-4 errors is given in Section 8.1.2.

In case of unsuccessful command execution, MF3E(H)x3 will execute the same abort actions as for native commands, see Section 7.5.3.

The following generic error cases can occur.

- ISO6985.: An ongoing wrapped chained command or multiple pass command is aborted, see Section 7.5.2.2.
- ISO6700: Wrong or inconsistent APDU length according to [10].
- ISO6D00: The received instruction code INS is not supported.
- ISO6A86: Incorrect parameters P1 or P2.
- ISO6581: In case of HW failure when reading or writing to non-volatile memory (not expected during normal operation). It is not mandatory to send this error code. The MF3E(H)x3 may also execute a reset.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.17.3.4 ISOSelectFile command

ISOSelectFile, as defined in compliance with ISO/IEC 7816-4 in in Section 8.12.1 and Figure 85 selects either the PICC level, an application, or a file within the application.

P1 defines the selection method.

If *P1* is set to 0x00, 0x01 or 0x02, selection is done by a 2-byte ISO file identifier. *P1* set to 0x00 can be used to select the MF (that is, the PICC level), a DF (i.e., a DESFire application if currently the PICC level is selected) or an EF (that is, a DESFire file within the currently selected application). *P1* set to 0x01 can be used to select a DF, if the MF is currently selected. *P1* set to 0x02 can be used to select an EF, if an application is currently selected.

For MF selection, 0x3F00 or empty data is to be used. For DF and EF selection, *Data* shall hold the 2-byte ISO/IEC 7816-4 file identifier as optionally set at application or file creation. Note the different byte order for the file identifiers when written with native commands and when used here for ISO/IEC 7816-4 selection.

If *P1* is set to 0x03, the MF level is selected. This option can only be issued if currently an application (DF) is selected. In this case, *Data* must be empty.

If *P1* is set to 0x04, selection is done by DF name which can be up to 16 bytes.

The registered ISO DF name of MIFARE DESFire itself is *0xD2760000850100*. When selecting this DF name, the PICC level (or MF) is selected. This is used as the default value for VCIID which is recommended to be changed to an installation-specific value. For selecting a DESFire application immediately, the ISO/IEC 7816-4 DF name, as optionally set at application creation is to be used.

ISOSelectFile.P2 shall indicate whether or not File Control Information (FCI) is to be returned in case of application selection.

MF3E(H)x3 supports three options for the FCI:

- if the targeted application is configured for integrated ECC-based authentication, see Section 7.5.6, while ECC-based Mutual and Reader-Unilateral Authentication is enabled for the targeted interface (NFC or I^2C), the FCI (if requested) holds the ephemeral public key of the PICC for the authentication.
- if the targeted application is configured for integrated ECC-based authentication, see Section 7.7.1.3, the FCI (if requested) holds the encrypted UID.
- if the above configurations do not apply, the FCI can hold static data from file 0x1F. P2 shall be set to 0x00 to retrieve this data, otherwise if 0x0C no FCI is returned.

In the latter case, FCI is returned as response data, if the following conditions are satisfied:

- the targeted application hold as file with native file number 0x1F.
- this file is of FileType.StandardData or FileType.BackupData.
- this file is freely accessible, that is, FileAR.Read or FileAR.ReadWrite holds the free access condition, see Section 7.12.2.
- *Le* is present.

The number of bytes requested by *Le* up to the complete file data will be returned in plain. There is no specific FCI template format checked, that is, the data stored in the file will be sent back as is. In case of PICC level or file selection, FCI data is never returned.

Otherwise, in case of integrated ECC-based authentication, Le must allow to return 69 bytes. If not, an error is returned.

Note that P2 set to 0xC is not supported by the VC selection command subset, and is therefore always handled by the DESFire VC that was already implicitly or explicitly selected.

PICC level (MF) or application (DF) selection aborts any current open transaction and starts a new one. It also aborts any active valid authentication and puts the card in VCState.NotAuthenticated state. In case of failure,

Provided under NDA only. CONFIDENTIAL - Personal copy for: Sergio Salata. Do not copy, share or reproduce. Quartis IT Solutions GmbH 5f07779d-92c4-405c-9066-6ecc3c8e4261

the current selection status both at application (MF/DF) and file (EF) level is not affected. The currently selected application and file, if any, remains selected.

When trying to select an application (DF) and the the TMCLimit of the targeted application was reached, see Section 7.14.3.2.2, the application will still be selected, but ISO6283 is returned to indicate limited functionality. This means CommitTransaction will be rejected, that is, no further transactions can be executed. Note that this is slightly different from pure MIFARE DESFire Light implementations, where all data management commands, except ReadData targeting the FileType.TransactionMAC file, will be rejected. Also CommitReaderID will be rejected. ISO6283 is sent together with the FCI.

Also, when the application holds a FileType.TransactionMAC configured for Transaction Signature, but the KeyID.AppTransactionSIGKey, that is, the referenced KeyID.ECCPrivateKey, does not exist or is not enabled for Transaction Signature, ISO6283 is returned and the same limited functionality applies.

7.17.3.5 ISOReadBinary command

ISOReadBinary, as defined in compliance with ISO/IEC7816-4 in Section 8.12.2 and Figure 86 can be used to read data from FileType.StandardData and FileType.BackupData files.

P1 and *P2* define the targeted file and the offset.

If Bit 7 of *P1* is set, then *P1* Bit 4-0 encodes a short ISO/IEC 7816-4 file identifier, that is, referencing the five least significant bits of the 2-byte ISO/IEC 7816-4 file identifiers. All zero bits are reserved for referencing the currently selected file. All one bits are reserved [10] and will be rejected. The referenced file will be selected for this and subsequent operations. Note that if intending to use short file identifiers, the user needs to take care of avoiding collisions amongst each other and with the reserved values in the definition of the file system, as there is no checking on file creation. *P2* encodes the offset from 0 bytes to 255 bytes.

If Bit 7 of *P1* is set, then *P1* Bit 6-0 concatenated with *P2* encode the offset from 0 to 32767 bytes. The file currently selected is targeted. If no file was selected, the command is rejected.

At PICC level (AID 0x000000), the command is rejected.

Le encodes the number of bytes to be returned. If the encoded value is 0x00 or if it is larger than the number of bytes in the file (starting from the offset), all remaining bytes of the file will be returned. In case of CommMode.MAC, the value encoded by *Le* must be strictly larger than the MAC size to be returned or 0x00.

As listed in Table 46, ISOReadBinary is allowed only if at least one of FileAR.Read, FileAR.ReadWrite and FileAR.SDMFileRead access rights associated with the targeted file is granted. It must be set to 0xE, that is, free access, as the command is only accepted in VCState.NotAuthenticated. Therefore, the communication mode of the targeted file is ignored.

Only Secure Dynamic Messaging is supported (which does not require a preceding authentication), depending on the targeted file's configuration, see section Section 7.5.11.

For FileType.BackupData, ISOReadBinary returns only data that has been written into the file and committed with a successful CommitTransaction in case of writing with wrapped WriteData. Before a successful commit, the old, unchanged data that was stored on the PICC will still be returned. Note that data written with ISOUpdateBinary is immediately committed.

The command is rejected if the application of the targeted file holds a FileType.TransactionMAC file, see Section 7.14.3.5.

The ISOReadBinary can be used to implicitly trigger an NFC Pause, as explained in Section 7.16.1.

Provided under NDA, ONLY CONFIDENTIAL Personal copy for: Sergio Salazar Quantum Solutions GmbH Do not copy, share or reproduce 5fc0779d-c2bc-4059-9bfe-6cc3bc6e42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.17.3.6 ISOUpdateBinary command

ISOUpdateBinary, as defined in compliance with ISO/IEC7816-4 in Section 8.12.3 and Figure 87, can be used to write data to FileType.StandardData and FileType.BackupData files. Updating a CRLFile, as defined in Certificate revocation is not supported with this command.

P1 and *P2* define the targeted file and the offset. The interpretation is identical as for ISOReadBinary, see Section 7.17.3.5.

At PICC level (AID 0x000000), the command is rejected.

Lc encodes the number of bytes to be written. The command is rejected if one attempts to write across the file boundary.

As listed in Table 46, ISOUpdateBinary is allowed only if at least one of FileAR.Write and FileAR.ReadWrite access rights associated with the targeted file is granted. It must be set to 0xE, i.e. free access, as the command is only accepted in VCState.NotAuthenticated. Therefore, the communication mode of the targeted file is ignored.

The FileType.StandardData does offer limited anti-tearing protection. It is ensured that the data received in a single frame is written anti-tearing protected, that is, all targeted data or none of it is updated. In case of chaining, see Section 7.5.2, the MF3E(H)x3 may buffer multiple frames and write them at once. Note that also in this case, if secure messaging applies, incomplete cryptographic blocks within a frame cannot be fully processed. Such a block will then be considered as part of the next frame.

Data written with ISOUpdateBinary is immediately committed, as if CommitTransaction is successfully executed afterward. Note that any other pending update within the currently selected application (via wrapped commands) is therefore also committed.

7.17.3.7 ISOReadRecord command

ISOReadRecord, as defined in compliance with ISO/IEC7816-4 in Section 8.12.4 and Figure 88, can be used to read data from FileType.LinearRecord and FileType.CyclicRecord files.

P1 and *P2* define the targeted file and record.

P1 encodes the record number. Value 0x00 is reserved for the current record, which means the last written record.

Otherwise, the actual record numbering is applied. Note that the ISO/IEC 7816-4 record numbering [10] is different from the native DESFire record numbering and differs between FileType.LinearRecord and FileType.CyclicRecord.

For FileType.LinearRecord, 0x01 is the first written (oldest) record. For FileType.CyclicRecord, 0x01 is the most recently written record. The command is rejected if no record or the targeted record has not been written yet.

P2 Bit 7-3 encodes a short ISO/IEC 7816-4 file identifier, that is, referencing the five least significant bits of the 2-byte ISO/IEC 7816-4 file identifiers. All zero bits is reserved for referencing the currently selected file. All one bits is reserved [10] and will be rejected. Note that if intending to use short file identifiers, the user needs to take care of avoiding collisions among each other and with the reserved values in the definition of the file system, as there is no checking on file creation. The referenced file will be selected for this and subsequent operations. *P2* Bit 2-0 encodes whether only the record targeted by *P1* is read or whether the record targeted by *P1* up to the last record is read.

Records are returned in the order of the numbering as defined by ISO/IEC 7816-4 [10]. This means also here that the order differs for FileType.LinearRecord and FileType.CyclicRecord.

At PICC level (AID 0x000000), the command is rejected.

Provided under NDA only - Personal copy for: Sergio Sanz
CONFIDENTIAL - Quantum Technologies GmbH
Do not copy, share or reproduce
5fc07f79c-c9ac-405c-90fe-6ec3ca0021

Le encodes the number of bytes to be returned. Note that it is possible that the last returned record is only partially returned. On the other hand, if the encoded value is 0x00 or if it is larger than the record size (in case *P2* Bit 2-0 is 100b) or larger than the size of the complete sequence of requested records (in case *P2* Bit 2-0 is 101b), all bytes of the single requested record or the requested sequence of records will be returned.

If *Le* is 0x00 and the number of requested bytes is bigger than 256 (normal Le) or, theoretically, 65536 (extended *Le*), an error will be returned. In case of CommMode.MAC, the value encoded by *Le* must be strictly larger than the MAC size to be returned or 0x00.

As listed in Table 46, ISOReadRecord is allowed only if at least one of FileAR.Read and FileAR.ReadWrite access rights associated with the targeted file is granted.

It must be set to 0xE, that is, free access, as the command is only accepted in VCState.NotAuthenticated. Therefore, the communication mode of the targeted file is ignored.

The command will also be rejected if the application of the targeted file holds a FileType.TransactionMAC file, see Section 7.14.3.5.

ISOReadRecord returns only data that has been written into the record file and committed with a successful CommitTransaction in case of writing with wrapped WriteRecord and UpdateRecord. Before a successful commit, the old, unchanged data that was stored on the PICC will still be returned. Note that data written with ISOAppendRecord is immediately committed.

7.17.3.8 ISOAppendRecord command

ISOAppendRecord, as defined in compliance with ISO/IEC7816-4 in Section 8.12.5 and Figure 89, can be used to write a new record to FileType.LinearRecord and FileType.CyclicRecord files.

P1 and *P2* define the targeted file and record.

P2 Bit 7-3 encodes a short ISO/IEC 7816-4 file identifier The interpretation is identical as for ISOReadRecord.

P2 Bit 2-0 and *P1* must be set to zero to indicate that a new record will be appended. For FileType.LinearRecord file, a new record is appended at the end of the file. This record will get the highest number in the ISO/IEC 7816-4 numbering. It is not possible to execute the command if the file is already full. For FileType.CyclicRecord file, the command appends a new record if not all records have been written or erases and overwrites the oldest record if the existing number of record equals the maximum number of records of the file. Note that the maximum number of records of a FileType.CyclicRecord file is one less than specified at creation. The new record will get number 0x01 in the ISO/IEC 7816-4 numbering.

At PICC level (AID 0x000000), the command is rejected.

Lc encodes the number of bytes to be written. The command is rejected if one attempts to write across the record boundary.

As listed in Table 46, ISOAppendRecord is allowed only if at least one of FileAR.Write and FileAR.ReadWrite access rights associated with the targeted file is granted.

It must be set to 0xE, that is, free access, as the command is only accepted in VCState.NotAuthenticated. Therefore, the communication mode of the targeted file is ignored.

The command will also be rejected if the application of the targeted file holds a FileType.TransactionMAC file, see Section 7.14.3.5.

The command is rejected if a wrapped native UpdateRecord or ClearRecordFile targeting the same file has already been performed during the current transaction.

Data written with ISOAppendRecord is immediately committed, as if CommitTransaction is successfully executed afterward. Note that any other pending update within the currently selected application (via wrapped commands) is therefore also committed.

Provided under NDA only
COMPANY CONFIDENTIAL. Personal copy for:
Sergio Garcia
Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fc07f9d-c22c-405c-97fe-6ec3caeb2ff

7.17.3.9 ISOGetChallenge command

ISOGetChallenge, as defined in compliance with ISO/IEC7816-4 in Section 8.12.6 and Figure 90, can be used as a random source compliant with AIS 31 DRG.3 [17].

P1 and P2 must be set to 0x00.

Le shall be set to to the requested length. Note that ISOGetChallenge can only be executed in VCState.NotAuthenticated and will not affect the authentication state on successful execution.

As ISOGetChallenge can only be executed with plain communication, one should be careful to execute only in a secure environment for, for example, key generation.

7.17.4 Selection of native frame versus ISO/IEC 7816-4 framing

After activation, MF3E(H)x3 will accept also a mixed sequence of native and ISO/IEC 7816-4 APDUs.

If the first command after ISO/IEC 14443-4 activation is an ISO/IEC 7816-4 APDU except ISOSelectFile targeting DF/MF, before execution of the first command, the application with native AID 0xEEE000 is selected if present. Within this application the file with native file number 0x00 is selected if present.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae4[illegible]

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.18 Proximity Check Architecture

The Proximity Check Reference Architecture allows the PCD to verify whether the PD is within a certain distance. The goal is to avoid the following attacks:

- A relay attack where an attacker tries to use a PD that is not in proximity to set up a transaction with a PCD. This attack can be undertaken with or without the support of the legitimate card holder.
- A relay attack where an attacker tries to obtain privacy-sensitive information from a card holder. The attacker tries to find out whether the attacked PD contains a VC belonging to a certain installation by relaying the requests of a valid PCD of that installation to a PD under attack. If the PD is accessed by the PCD after a successful VC Selection procedure, the attacker knows the PD also belongs to the installation, which may reveal for example that a card holder is a London citizen (that is, has an access card for the London Subway).
- A DPA attack on the actual keys of a VC. When proximity checking is not mandatory, the actual keys of a VC can be accessed without limitation and many traces of failed authentications can be made (depending on whether authenticated VC selection is mandatory, with or without relaying to a valid reader). Mandating Proximity Checking by the PD forces an attacker who wants to make many DPA traces to carry out this process in proximity of the legitimate PCD. That introduces another barrier.

The Proximity Check is an optional, but highly recommended step after the (implicit or explicit) selection of a VC, as it increases the security by averting the attacks described above. MF3E(H)x3 also supports it once the card has been selected, as described in Section 7.18.4. As it is only an optional step, this specification leaves some freedom to the PCD in how to implement it (for example, choosing an appropriate threshold time).

The Proximity Check is performed by the PCD on the PD. There is no need for a Proximity Check performed by the PD on the PCD because both the PD and PCD are part of a closed system managed in one installation and therefore the PD needs to trust the PCD anyway. For proximity checking the PD trusts a valid PCD to correctly perform the Proximity Check. However, when it has shown to possess the right key material, the PD will get a confirmation from the PCD at the end of the protocol.

Note: *MF3E(H)x3 does not support Proximity Check over the I^2C interface.*

7.18.1 Technological basis for proximity checking

The proximity checking is carried out by measuring the round-trip time of a challenge-response interaction. If an attacker wants to mount a relay attack, then he will necessarily introduce delays. Depending on how large the delays are, they may be detected. The accuracy of the time measurement and the residual relay attack window that remains, is dependent on the implementation of the PCD's Contactless Front-End (PCD-CLF), that is, the part of the PCD that takes care of the contactless communication, as well as the PD implementation. Relay attacks that are mounted over a short distance using a high-bandwidth communication will not be detectable. However, mounting a relay attack by using communication channels with significant delay, for example, mobile phones, can be countered by this implementation.

Although in theory the check could be done with a single command and single response, this protocol uses at least three command-response pairs. The actual drawing of a random number at the PD side and the cryptographic calculation may take more time than the time within which the response must be returned.

Therefore those three elements are decoupled into:

1. Let the PD draw a random number and let the PD not send a response before it is done.
2. Answer with the response-random number once the challenge-random number has arrived. The PCD can split this step into multiple challenge-response pairs.
3. Perform the cryptographic check to make sure that the numbers have not been tampered with.

The Proximity Check set-up and its goal are illustrated in Figure 23. Note that in case of a PICC the PD-CLF is integrated.

Provided under NDA only
COMPANY CONFIDENTIAL
Serial: Sata Solutions GmbH
Quantity: 1 Personal copy for:
Do not copy, share or reproduce
5fc07700-c2c-40ec-90fe-6ecc2ce4214

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

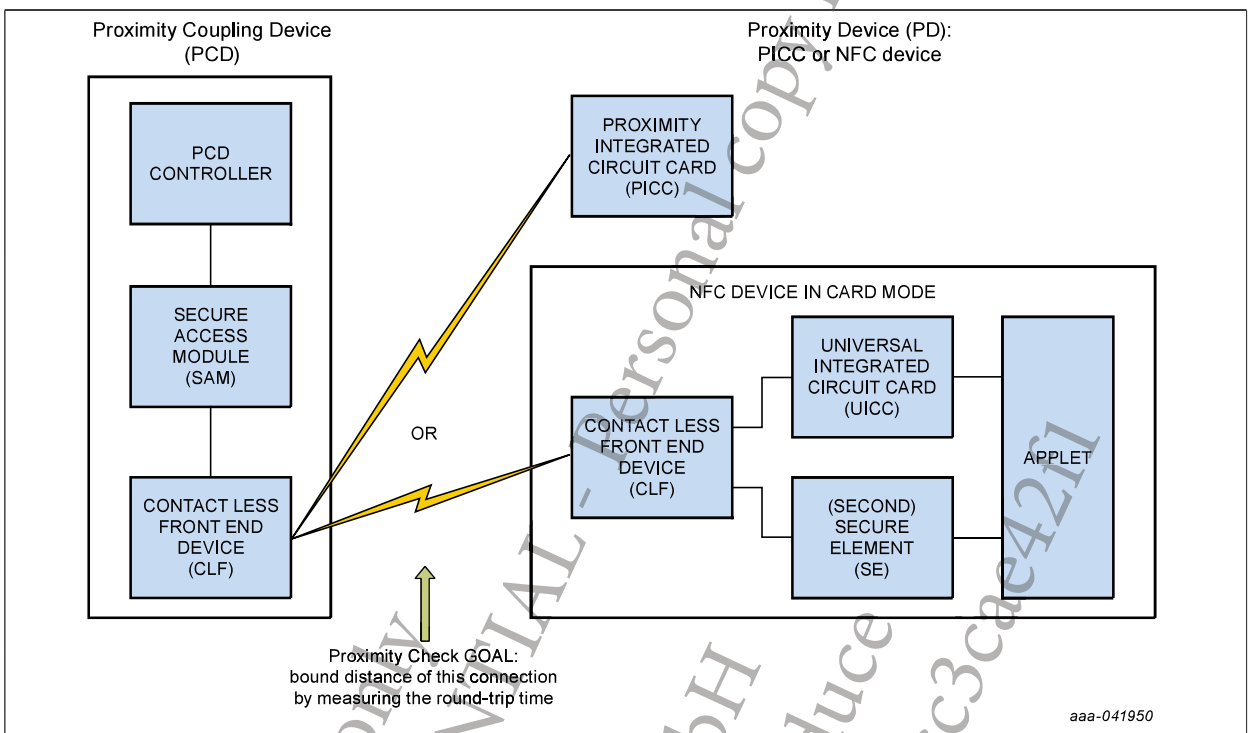


Figure 23. Proximity Check set-up

7.18.2 Proximity Check protocol overview

The Figure 27 gives an example of the messages exchanged during a Proximity Check.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90f6-6ecc3cae4211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

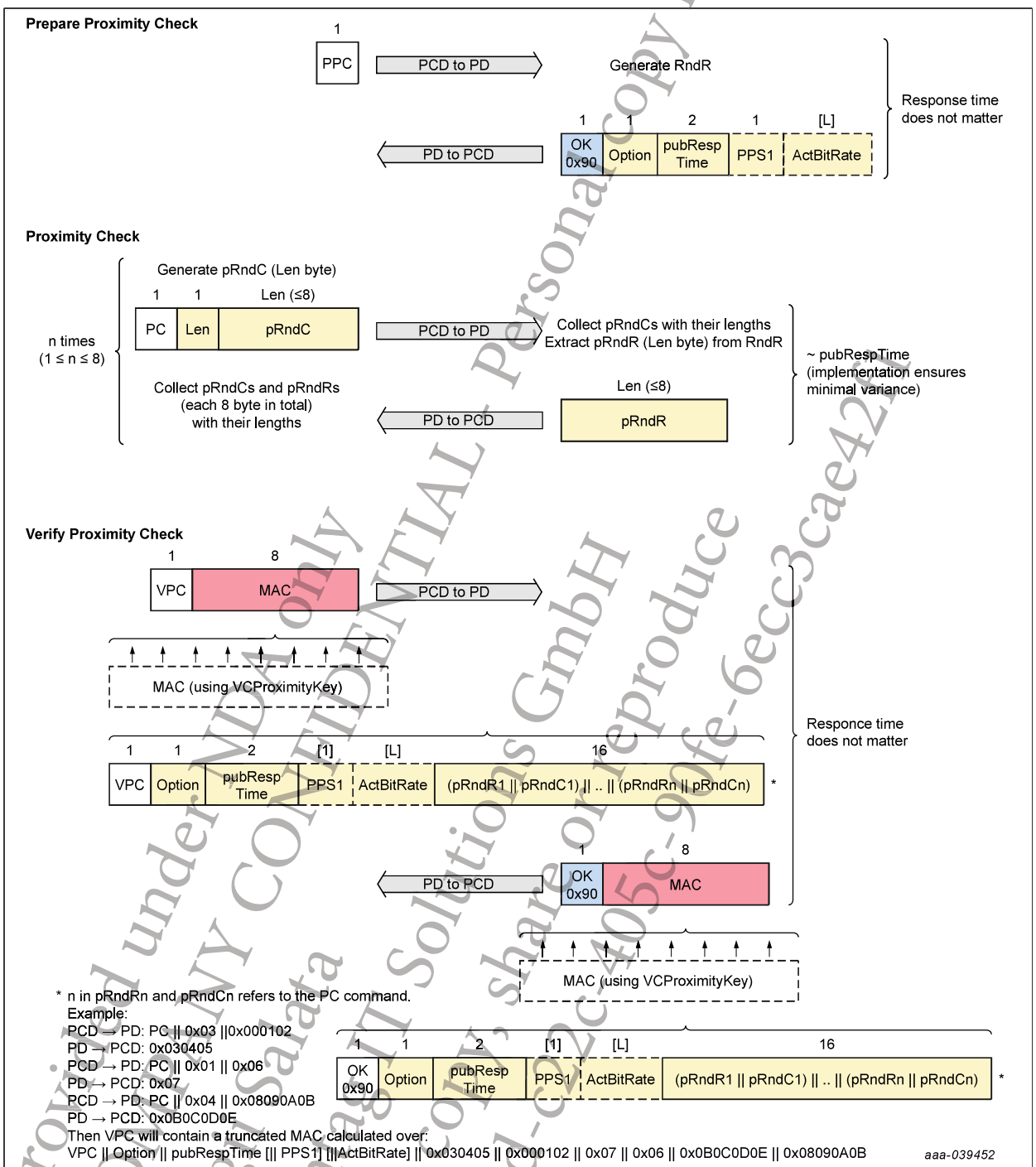


Figure 24. Proximity Check messages

If the Proximity Check protocol fails, either because of exceeding the threshold time or because of a failed MAC verification (see Section 7.18.2.2 and Section 7.18.2.3 for details), it is recommended that a PCD aborts the transaction (for example, by executing an ISO 14443-4 DESELECT command). A PCD may abort the transaction after the complete execution of the Proximity Check protocol, but not before it can reasonably have

Provided under NDA only
COPY ONLY CONFIDENTIAL Personal copy for:
Sergi Satata
Do not copy, share or reproduce
Quantum IT Solutions GmbH
5fc0779d-92c-405c-90fe-6ecc3cae42fa

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

checked the VerifyPC response. Best is to wait until the moment that otherwise an Authenticate command would be sent.

When detecting that the timing is violated (for example, after the first Proximity Check challenge-response pair) the PCD can stop interacting, as an attacker cannot obtain information from such PCD behavior regarding the requested VCIID being supported by the PD.

When an unexpected command is received during the execution of the Proximity Check protocol, the PD must stop the protocol. The received command will be rejected and the interrupted Proximity Check protocol cannot be finished successfully anymore.

This allows for easy recovery from communication errors. For example, when the PCD discovers a communication error in one of the PC commands (for example, by receiving an incorrect CRC), it can immediately initiate the protocol again by sending a new PreparePC. This will be accepted by the PD, in the meantime also stopping the previous protocol. Note that this is not without risk, as it allows an attacker to increase the number of trials for a guessing attack by inserting communication errors once he discovers a wrong guess. For this reason, it is recommended that a PCD puts an upper limit on the number of retrials it executes due to communication errors.

Note that in case of communication errors in a ProximityCheck, detected at PD side, normal ISO/IEC 14443-4 protocol handling (via R-blocks) may lead to retransmission of the PC command. In this case, the PCD should not restart the timer, effectively leading to a time-out in general. Also during ProximityCheck a PCD should not accept an S(WTX) according to [6]. This should also lead to a failure. Note that if re-configuring the ATS, the user should be careful not to trigger S(WTX) sending by configuring the FWI to a too low value.

When the PCD needs to abort and restart a proximity check, the PCD must draw a new random challenge.

To limit the residual relay window for an attacker, the response time of the ProximityCheck should be as small as possible. Preferably it is close to the earliest that ISO 14443-4 allows: that is, after the minimal Frame DelayTime. Note that this is specified in [ISO 14443-3]. The minimal Frame Delay Time slightly varies depending on the different data rates used in ISO 14443-4. See Table 2 in [5].

However, not every implementation will be able to achieve this. Consider when the PD is an NFC device in card mode. Then the PD consists of a CLF and an SE. Normally, the CLF only receives the commands and forwards them to the SE. It is clear that this will require more processing time at the PD for the actual ProximityCheck.

For this reason, the PD will indicate its expected response time within the PreparePC response. Based on this value, the PCD can derive an appropriate threshold time for its timing measurement and this way handle a mixed population of PD implementations. A PD implementation shall guarantee that it will not exceed its published response time.

Note that the residual relay attack window does not directly depend on the size of the expected response time of the ProximityCheck, but rather on the difference between the actual response time of the PD and the expected response time by the PCD. This difference, which could in general vary due to implementation and execution variations or due to attacker manipulation (for example, by increasing the RF frequency), is to be kept as low as possible by a PD implementation.

The PD implementation guarantees that it will not exceed the published response time but also that it will not deviate at the lower bound from it by more than 20%. So the PD must also not send the response sooner than this limit, as this will directly increase the residual relay attack window available for a relay attacker.

7.18.2.1 PreparePC command

First the PCD sends a PreparePC, as defined in Section 8.13.1 and Figure 91. This instructs the PD to prepare a 8-byte random number *RndR* (Random Response).

The PreparePC only consists of a command code.

Provided under NDA only - Personal copy for: CONFIDENTIAL - Serial number: Quantum TT Solutions GmbH - Do not copy, share or reproduce - 5fc07779d-92c-4952-9bf8-6ec3c4421

In case the PICC level KeyID.VCProximityKey is targeted but still disabled, see Section 7.8, the PreparePC is rejected. It is answered by the PD with OK. This is followed by *Option*, indicating whether the *PPS1* byte or *ActBitRate* is present, *PubRespTime* and, optionally, *PPS1* or *ActBitRate*.

If the bit current bit rate is returned, a PD shall use the *PPS1* encoding if possible, that is, if still running at a bit rate supported by *PPS1* even if it has been activated by the S(PARAMETER) support of [6]. Note that this does not necessarily mean the *PPS1* was exchanged during the ISO/IEC 14443-4 activation when *PPS1* equals 0x00, that is, both sides operating at the default bit rate. Note that *PubRespTime* is encoded MSB first.

Note that if the bit rate is still updated after PreparePC, the protocol will fail, that is, subsequent ProximityCheck commands operating at a different bit rate than what was published on PreparePC will be rejected.

Note that *PubRespTime* is encoded MSB first.

7.18.2.2 ProximityCheck command

After PreparePC, the PCD sends a ProximityCheck (PC) with a part of or the full 8-byte random challenge *RndC* (Random Challenge), as defined in Section 8.13.1 and Figure 91. The PCD may send the whole 8-byte number at once, down to only one byte at a time.

The ProximityCheck consists of the 1-byte command code *cmd*, a 1-byte length field *Len* and a random number *pRndC* (with a length between 1 byte and 8 bytes).

The command is transmitted compliant to the ISO/IEC 14443-4 layer, including prologue (PCB and optional CID) and epilogue (CRC) fields [6]. Directly after sending the last bit, it is recommended that the PCD starts a timer.

Then the PD responds *pRndR*, that is, a part of its prepared random response of equal length.

It is sent as close as possible to the time published on PreparePC.

The ProximityCheck response is transmitted compliant to the ISO/IEC 14443-4 layer, but it does not contain a return code in case of success to limit the residual relay attack window.

It is recommended that the PCD stops the timer directly when it receives the first bit of the PD response. The measured time is the time between sending the random number and receiving the responding random number, that is, the actual response time (which should be as close as possible to the published response time) plus the Round-Trip Time (RTT).

If the PCD would normally accept subsequent bits of the response coming in at times which are later than what the standard [6] prescribes, the PCD must also measure the time until the last bit arrives. Otherwise an attacker could guess a first bit with 50 % probability and then send the relayed bits later when available.

This procedure of sending part of the random numbers as a challenge-response pair is repeated by the PCD until the complete 8-bytes are exchanged. In what sizes the random number is split up and, consequently, how many Proximity Check commands are used by the PCD can be chosen freely and is to be specified by the Contactless Infrastructure Operator. Obviously, the maximal number of challenge-response pairs can be seven. Both ends collect the received parts of the 8-byte random number and remember the order in which they were received, as this is needed for the final command of the protocol.

If the current bit rate was published on PreparePC, and the ProximityCheck is executed at a different bit rate than what was published due to bit rate change in between, the ProximityCheck will be rejected.

The measured time is compared against a predefined threshold. The threshold value is unspecified by this document. It is to be chosen by the Contactless Infrastructure Operator depending on the granularity of time measurements possible with the PCD-CLF and an evaluation of the residual risks. A PCD can check the measured time after each Proximity Check command (or use a timer that times-out when exceeding the threshold value), or it can remember the maximal time measured and do the check only at the end. If the threshold value is exceeded, the Proximity Check protocol fails.

Provided under NDA only
CONFIDENTIAL - Personal copy for:
Georgi Stanev
Giesecke+Devrient GmbH
Document Portal
Date of request: 2025-07-10
5f0f7f9d-0002-4000-90fe-6ec3ca4251

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

It is recommended that a PCD aborts the transaction in this case by performing a Deselect or removing the power from the field. A PCD may also delay this abort until the end of the protocol.

7.18.2.3 VerifyPC command

Once the complete 8-byte random challenge was processed with the Proximity Check commands, the PCD sends a VerifyPC (VPC), as defined in Section 8.13.3 and Figure 93.

The VerifyPC consists of the 1-byte command code *cmd* and the 8-byte MAC *MAC*. The *MAC* is taken over the complete 8-byte random numbers and the information published by the card on the PreparePC response, so that a man-in-the-middle attacker cannot manipulate this information and extend the published response time or operate the card at a different (higher) speed and gain time to mount an attack. The random number ordering for the MAC input reflects the same split as during the sending of the Proximity Check commands. The complete input is: *0xFD* || *Option* || *pubRespTime [*|| *PPS1] [*|| *ActBitRate]* || *pRndR1* || *pRndC1* || ... || *pRndRn* || *pRndCn*. So the *PPS1* byte or *ActBitRate* is only included in the calculation if it was published in the response of the PreparePC. The PD will verify the incoming MAC.

If the current bit rate that was published on PreparePC changed in the meantime, the VerifyPC will be rejected.

If MAC verification succeeded, the PD sends as response MAC that is calculated over the same information, replacing the command code by the OK return code. The complete input is: *0x90* || *Option* || *pubRespTime [*|| *PPS1] [*|| *ActBitRate]*|| *pRndR1* || *pRndC1* || ... || *pRndRn* || *pRndCn*. If it failed, the PD will replace the MAC with a random number. When receiving this response, the PCD verifies the MAC. If the MAC verification fails, the Proximity Check protocol fails; otherwise it was successful.

It is recommended that a PCD aborts the transaction in case of failure by performing a Deselect or removing the power from the field.

The key to be used for these MAC operations depends on when the Proximity Check is performed throughout the complete VC control flow. This is specified in Section 7.18.4.

7.18.3 ISO/IEC 7816-4 support

MF3E(H)x3 enables ISO/IEC 7816-4 compliant Proximity Check by supporting wrapping of the native Proximity Check commands into ISO/IEC 7816-4 compliant APDUs, as defined in Section 7.17.2.

All PC related commands can be wrapped: PreparePC, ProximityCheck, and VerifyPC. Note however for the ProximityCheck that the wrapped variant will hold the status words *SW1*||*SW2* set to 0x9190. As they come at the end of the response, they do not add to the residual relay attack window like a native response code would do.

Extended length is not supported for proximity commands.

7.18.4 Proximity Check location and keys

This section describes when a PCD can and/or should execute a Proximity Check. This section lists all possible locations in the global command flow and describes which keys must be used for each of them.

- If not authenticated, no AES Session Keys are available, so the KeyID.VCProximityKey must be used for the VerifyPC responses.
- If the VC has successfully been selected and is authenticated, that is, both in VCState.AuthenticatedAES and VCState.AuthenticatedECC, since an AES session key (for MACing) is available, this key KeyID.SesAuthMACKey must be used for the VerifyPC responses.
- During any ongoing command, a PD must not support Proximity Checks. Any command of the Proximity Check protocol is treated as any other unexpected command.

Provided under NDA only. Personal copy for: COMPANY CONFIDENTIAL Sebastian Salata, IT Solutions GmbH Do not copy, share or reproduce. Q-Portal-id: 5fc07f2d-c92c-4035-90fe-6ecc3ca4ffff

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

By default, the KeyID.VCProximityKey is the PICC level key. If an application is selected, the application-specific key (if any was created at application creation time) is used. If two applications are selected, only the application level key of the first application is considered.

In case the PICC level KeyID.VCProximityKey is targeted but still disabled, see Section 7.8, the PreparePC is rejected.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.19 EV Charging

To allow integration into EV Charging systems, MF3E(H)x3 supports the command set and Trust Provisioning configuration options defined by the German standardization committee VDE-DKE in VDE-AR-E 2532-100 "Requirements for an authentication for the use of electric mobility supply systems" [33].

The standard defines a unilateral card authentication mechanism using an ECDSA-based challenge-response protocol supported by a certificate. The targeted application will hold the private key (KeyID.ECCPrivateKey) from a key pair on the brainpoolP256r1 curve. Next to this, there are two files: one holding a related certificate, and a second file for additional data. The private key and certificate can be preprovisioned during manufacturing as required by the standard.

VDE-AR-E 2532-100 specifies four commands, which are either existing MF3E(H)x3 commands or wrappers around MF3E(H)x3 functionality already described in the previous chapters.

SELECT for selecting the application. This command is a subset of the functionality of ISOSelectFile, see Section 7.17.3.4. How the further EV Charging commands can be enabled for a targeted application is explained in Section 7.19.1.

ReadData for reading out the certificate and potentially additional card holder data. This is implemented by VDE_ReadData, see Section 7.19.3.1.

ECDSASign for executing the unilateral card authentication. This is implemented by VDE_ECDSASign, see Section 7.19.3.3.

WriteData for writing additional information to the additional file, for example, during in the field enrolling. With this command, this file can then also be locked to avoid any further writing to it. This is implemented by VDE_WriteData, see Section 7.19.3.2.

7.19.1 Enabling EV Charging support

The EV Charging functionality, as required by VDE-AR-E 2532-100 [33], can be enabled for an application by issuing SetConfiguration Option 0x04 enabling Bit 4 of SMConfig, see Section 7.7.3.1 and especially Section 7.7.3. Enabling the EV Charging functionality for an application means in practice that the three commands specified in Section 7.19.3 (VDE_ReadData, VDE_WriteData and VDE_ECDSASign) are supported by that application. In applications where the EV Charging functionality is not enabled, these commands are rejected.

VDE-AR-E 2532-100 specifies that the EV Charging application must have the ISO/IEC 7816-4 AID (that is, DF Name) "A0000008450000000000000000000001" for selection with ISOSelectFile. MF3E(H)x3 supports enabling the functionality in any application, therefore it must not necessarily have the above DF Name.

The functionality is only supported at application level, that is, not at PICC level.

7.19.2 Data structures

To execute the EV Charging commands specified in the next section successfully, the following data structures must be present in the targeted application:

- An KeyID.ECCPrivateKey with KeyNo 0x00. VDE-AR-E 2532-100 requires the key to be on brainpoolP256r1 curve, but MF3E(H)x3 also supports NIST Curve P-256.
- A FileType.StandardData file with FileNo 0x00, holding the certificate. Note that MF3E(H)x3 is agnostic about the actual content of the file, that is, whether the file holds a valid certificate does not affect the MF3E(H)x3 processing.
- A FileType.StandardData file with FileNo 0x01, holding additional data. Also here MF3E(H)x3 is agnostic about the actual content.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Service Solutions GmbH
Quantum IT Solutions
Do not copy, share or reproduce
5fc07760-c22c-405c-bfe-6ec3cae2210

MF3E(H)x3 supports delivery options where the application and these files and key are preprovisioned to the card during manufacturing. This is further outlined in Section 7.22 on trust provisioning support.

7.19.3 Command support

The EV Charging commands defined in this section are ISO/IEC 7816-4 APDU from the proprietary class, that is, they have their CLA byte set to 0x80.This is different from the interindustry commands with CLA byte 0x00, see Section 7.17.3, but also from the wrapped native APDUs, see Section 7.17.2, which are also part of the proprietary class, but have their CLA byte set to 0x90.

If EV Charging functionality is not enabled for the currently selected application, the commands defined in this section will be rejected.

Next to the commands defined in this section, VDE-AR-E 2532-100 also requires support of ISOSelectFile for application selection, see Section 7.17.3.4. If two applications have been selected with SelectApplication, the commands in this section target files and keys in the primary application.

7.19.3.1 VDE_ReadData

The command VDE_ReadData, as defined in Section 8.14.1 and Figure 94, supports retrieving the data of FileNo 0x00 or 0x01. The targeted file is specified by P1.

The targeted file must be a FileType.StandardData file that does not have SDM enabled, and is not a CRLFile. The command is also rejected if the application of the targeted file holds a FileType.TransactionMAC file, see Section 7.14.3.5.

As listed in Table 46, VDE_ReadData is allowed only if at least one of FileAR.Write and FileAR.ReadWrite access rights associated with the targeted file is granted. It must be set to 0xE, that is, free access, as the command is only accepted in VCState.NotAuthenticated.

Internally, the command functionality is equivalent to a ReadData with its FileNo set to the value in P1, that is, 0x00 or 0x01, and Offset and Length set to 0x000000. In case of short Le, only up to 256 bytes are returned, that is Length would be set to the minimum of the file size of the targeted file and 256 bytes.

7.19.3.2 VDE_WriteData

The command VDE_WriteData, as defined in Section 8.14.2 and Figure 95, supports writing data to FileNo 0x01, and eventually locking the file once all data is written.

P2 specifies the targeted operation. The value 0x00 triggers a write operation, writing the provided data to the FileType.StandardData with FileNo 0x01. The value 0x01 locks the file for any further writing.

The targeted file must be a FileType.StandardData file that does not have SDM enabled, and is not a CRLFile. The command is also rejected if the application of the targeted file holds a FileType.TransactionMAC file, see Section 7.14.3.5.

As listed in Table 46, VDE_WriteData targeting a write operation is allowed only if at least one of FileAR.Write and FileAR.ReadWrite access rights associated with the targeted file is granted. When targeting the lock operation, the FileAR.Change must be granted. This means that the relevant access right must be set to 0xE, that is, free access, as the command is only accepted in VCState.NotAuthenticated. FileAR.Read is not affected by this command: to allow further reading with VDE_ReadData, it is required to have FileAR.Read already configured to 0xE.

When writing to the file, the provided data will be written at the start of the file, that is, with offset zero. When locking, this means that all FileAR.Write and FileAR.ReadWrite, including any additional access rights (if any, see Section 7.12.2), and the FileAR.Change will be set to 0xF.

Personal copy for: Bergmann, Malaa
Do not copy, share or reproduce
Contact ITS Solutions GmbH
COMPANY CONFIDENTIAL - Personal copy for:
5fc07798-2e26-45c-90f2-66cc3cae42f1

Internally, when writing, the command functionality is equivalent to a WriteData with its FileNo set 0x01, Offset set to 0x000000 and Length set to the value in Lc. When locking, the command can be mapped to ChangeFileSettings with FileNo 0x01, all FileAR.Read, FileAR.ReadWrite and FileAR.Change set to 0xF while all other parameters keep their current value.

7.19.3.3 VDE_ECDSASign

The command VDE_ECDSASign, as defined in Section 8.14.3 and Figure 96, supports signing a 32-byte challenge, as specified by VDE-AR-E 2532-100 [33].

The command implements an ECDSA Digital Signature Generation as defined in [28]. The supported hash function is SHA-256, as specified in NIST FIPS 180-4 [25].

Similarly as CryptoRequest, VDE_ECDSASign requires the access conditions as set in the CryptoAPI Management (see SetConfiguration Option 0x15). As VDE_ECDSASign is only accepted in VCState.NotAuthenticated, the access condition must be set to 0xE and the communication mode is ignored.

The key targeted VDE_ECDSASign is the KeyID.ECCPrivateKey with KeyNo 0x00. This key must have ECC Sign operations enabled, as configured with Bit 4 of the associated KeyPolicy, see ManageKeyPair. While VDE-AR-E 2532-100 requires the key to be on brainpoolP256r1 curve, MF3E(H)x3 also supports NIST Curve P-256. If the key has an enabled KeyUsageCtrLimit that was already reached, see Section 7.9.1.2, the command is rejected.

Internally, the command functionality is equivalent to a CryptoRequest_ECCSign command with the following parameters:

- Action = 0x03, that is, ECC Sign.
- KeyNo = 0x00, that is, fixed mapping to the KeyID.ECCPrivateKey withKeyNo 0x00.
- Algorithm = 0x00, that is, ECDSA with SHA256.
- InputSource = 0x00, that is, RFU.
- InputType = 0x01, that is, Raw data to be hashed.
- InputLen = 0x20, that is, fixed to 32 bytes.
- InputData = Data from VDE_ECDSASign, that is, the 32 random challenge The response is the 64-byte signature consisting of (r,s).

7.19.4 Trust Provisioning of EV Charging Data Structures and Key Material

As explained in detail in Section 7.22, MF3E(H)x3 allows trust provisioning of applications and data structures as well as key and certificate material into the IC, directly during the NXP manufacturing process.

On selected product configurations, the EV charging application will be pre-installed with a trust-provisioned key pair and certificate.

7.19.4.1 EV Charging application

When the EV Charging application is pre-installed, it will come with the following configuration.

The mentioned configuration including application structure, file structure, and keys with related certificates is present in the dedicated MIFARE DUOX EV Charging part types. Ordering information for the MIFARE DUOX EV Charging part types is available in Section 4.2.

In addition to the trust-provisioned key pair and certificate, a defined number of symmetric keys are created in the application, to support additional functionality (e.g. creating additional files, or writing to file 0x0 after authentication).

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3ca2f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

- AID: 0x1010F6 (MSB first representation)
- KeySett1 (AppKeySettings): 0x09, i.e.:
  - Change key access right: 0x0.
  - AppKeySettings changeable with KeyID.AppMasterKey authentication.
  - No free file creation/deletion.
  - No free file directory access.
  - KeyID.AppMasterKey changeable with KeyID.AppMasterKey authentication.
- KeySett2: 0xB5, i.e.:
  - KeyType.AES128.
  - ISO/IEC 7816-4 File Identifiers enabled.
  - KeySett3 enabled.
  - Five application keys, initialized to to the default value of all zero bytes for key value and version.
- KeySett3: 0x04, i.e.:
  - Application deletion depending on PICCKeySettings.
  - Application-specific capability data enabled, i.e. for CryptoAPI configuration (see Section 8.3.3 Option 0x15).
  - No KeyID.VCProximityKey.
  - No key sets.
- ISOFileID: 0x00DF.
- ISODFName: 0xA0000008450000000000000000000001 (as defined by [33]).
- Section 8.3.3 Option 0x04 SMConfigB: 0x10, i.e. enable EV Charging.
- Section 8.3.3 Option 0x15 CryptoAuthAC: 0x0E, i.e. free access to VDE_ECDSASign.

Note that as five KeyID.AppKeys (including the KeyID.AppMasterKey) are initialized with the default byte value of all zero on delivery, the customer shall update those keys before handing out the card to avoid abuse in the field.

7.19.4.2 EV Charging KeyID.ECCPrivateKey entry

The EV Charging application will hold an KeyID.ECCPrivateKey entry

- Unique key pair per die, that can be used for EV Charging functionality.
- The private key is stored as KeyID.ECCPrivateKey KeyNo 0x00, with following default configuration:
  - brainpoolP256r1
  - KeyPolicy 0x0010, only allowing ECC Signing. Note that for allowing VDE_ECDSASign in VCState.NotAuthenticated the access condition for Crypto API Management CryptoAuthAC (as in general configurable with Cmd.SetConfiguration Option 0x15) shall be set to 0xE for this application.
  - WriteAccess: 0x0F, disabling update.
  - KUCLimit: disabled.
- The public key is defined and trusted via the certificate stored in File 0x00, see Section 7.19.4.3 and Section 7.19.4.4.

7.19.4.3 EV Charging Files

The EV Charging application will hold two files.

A first file holding the certificate is a FileType.StandardData file of 193 or 199 bytes (respectively for 7-byte or 10-byte UID) with following properties:

- FileNo = 0x00; ISO File ID = 0x00EF
- FileAR.Read = 0xE; FileAR.Write = 0x0; FileAR.ReadWrite = 0x0; FileAR.Change = 0x0.
- Secure Dynamic Messaging and mirroring are not supported for this file.

Provided under NDA only
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ec63cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

- Additional access rights are not supported for this file.
- CommMode.Full.

This file will hold the EV Charging Certificate as defined in Section 7.19.4.4.

A second file holding the certificate is a FileType.StandardData file of 1024 bytes with the following properties:

- FileNo = 0x01; ISO File ID = 0x01EF
- FileAR.Read = 0xE; FileAR.Write = 0xE; FileAR.ReadWrite = 0xE; FileAR.Change = 0xE.
- Secure Dynamic Messaging and mirroring are not supported for this file.
- Additional access rights are not supported for this file.
- CommMode.Plain This is not relevant, as only free access is supported.
- This file will be filled with zero bytes on delivery.

7.19.4.4 EV Charging Certificate

As defined above, file 0x00 will hold the EV Charging Certificate over the public key of the application's key pair, signed by a certificate authority (CA) maintained by NXP. This CA will be dedicated for the EV Charging application on MF3E(H)x3. As this application will be pre-installed on a standard delivery type, it means that the CA is shared across different customers. Therefore, the customer will need to allowlist instances before adopting them in their EV Charging system. Alternatively a customer can order a customized delivery type with a customer-specific CA maintained by NXP for the customer.

The CA maintained by NXP dedicated for the EV Charging application is identifiable via the CA ID 63709320010002.

Certificates of this EV Charging CA can be retrieved via the following link: https://www.gp-ca.nxp.com/CA/getCA?caid=63709320010002

The certificate has a GlobalPlatform structure as defined by [33], with following specifics:

Table 48. EV Charging Certificate

Tag	Length	Value	Description
0x7F21	179 or 185	-	Certificate
0x93	7 or 10	UID	Certificate serial number encoding the 7-byte or 10-byte UID
0x42	7	CA-ID	CA Identifier holding BCD-encoded 14-digit CA-ID 63709320010002 allowing retrieval from https://www.gp-ca.nxp.com/CA/getCA?caid=63709320010002
0x5720	7 or 10	UID	Subject Identifier holding the 7-byte or 10-byte UID
0x95	2	0x02 0x00	Key usage indicating digital signature
0x5F25	4	YYYMMDD (BCD format)	Effective date, holding production date
0x5F24	4	YYYMMDD (BCD format)	Expiration date, holding production date + 20 years
0x45	1	0x00	CA Security Domain Image Number, fixed to 0x00 by [33]
0x7F49	70	-	Public key data object, see Table 49.
0x5F37	64	r || s	Signature: ECDSA with SHA-256 signature (without additional encoding). The signed data include the tags 0x93 until and including 0x7F49.

Provided under NDA only. CONFIDENTIAL. Personal copy for: COMPANY CONFIDENTIAL Sergii Salata Quantum IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-4f5c-90fe-6ecc3c4a2f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 49. EV Charging Certificate Public Key

Tag	Length	Value	Description
0xB0	65	0x04||Pub.x||Pub.y	ECC Public Key in uncompressed point representation
0xF0	1	0x03	Key Parameter Reference: brainpool P256r1

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.20 Upgradeability architecture

Features within the MF3E(H)x3 product architecture are designed to have a long life. This means that they must be upgradable, either to correct errors, deal with vulnerabilities that may emerge over time or provide new features. This chapter describes the architectural provisions that are designed in to take care of upgradeability.

Some of the upgrade requirements that are foreseen are worked out both as a test of the concept and for direct use when those new requirements have to be supported.

7.20.1 Capability negotiation

The prime architectural concept that enables upgradeability is capability negotiation.

In capability negotiation, the PD and the PCD exchange their capabilities and then, in general, take for each capability the highest capability that they have in common. What happens in detail is that the PCD presents its capabilities first and then the PD presents its capabilities with as maximum the PCD capabilities, or its minimum required capabilities. Further details can be found in Section 7.20.4.2.3.

Some capabilities are combined into capability sets. Such capability sets may be on a high abstraction level like *MIFARE DESFire release 5*, defining in one capability set all the capabilities of a certain release.

For each capability or capability set it will be defined how it will be interpreted and dealt with in the capability negotiation.

Both PD and PCD can have minimum capabilities. An entity that is presented capabilities of the other entity that are below its own minimum capabilities may stop at a certain point in the protocol. This point must be chosen such that no side-channel information is leaked which could affect privacy.

For every capability or capability set it must be defined how the minimum value is to be interpreted.

7.20.2 Upgrading the MIFARE DESFire EV2 Secure Messaging Protocol

Upgrades to the Secure Messaging protocol may be needed over time. For example, a different way to generate session keys, or the ability to encrypt the commands. Such upgrades can be negotiated via the capabilities in the authentication protocol.

Further, the AuthenticateEV2First command includes a length field. As long as the number of bytes received does not conflict with the length field, it must not be rejected because bytes are received of which the meaning is as yet not clear. Later versions of the architecture could use more or longer operands and accepting a larger message with potentially unknown content makes that upgraded PCDs will not break non-upgraded PDs.

7.20.3 Upgrading the ECC-based mutual and reader-unilateral authentication

Upgrading the ECC-based mutual and reader-unilateral authentication, see Section 7.5.6, and subsequent secure messaging is possible via the *OptsA* and *OptsB* parameters exchanged in the protocol. Therefore, the PD already accepts bigger *OptsA* TLV-structures, ignoring the additional bytes.

If a different or upgraded authentication protocol is needed, then this may need to be communicated by the PD via the capabilities in the FCI of ISOSelectFile. This is not defined in detail for now, but could be done via adding a Tag 0x86 (Security attribute in proprietary format) as defined by [11]. Especially for the flow initiated with ISOSelectFile, see Section 7.5.6.6. Additionally, a Tag 0x86 may be added at some point to indicate upgrade.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergey Sadakov, Quarkslab SAS
Do not copy, share or reproduce
5fc07790-0227-4035-90de-6ecc3cae9211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.20.4 Coding of capabilities and detail behavior

7.20.4.1 Capability exchange during virtual card selection

7.20.4.1.1 PD capabilities

The PD capability information consists of 1 info byte and 2 capability bytes as shown in Table 50.

Info	PDCap1.1	PDCap1.2

Table 50. PD capability information

Capabilities Bytes	Information Field	Value	Description
Info			Chipmaker Defined Capabilities.
	VCUIDL		VCUID Length
		10XX XXXX	4-byte VCUID
		00XX XXXX	7-byte VCUID
		01XX XXXX	10-byte VCUID
		11XX XXXX	RFU
	VCTID		VC Type Identifier
		XX00 0000	Reserved. If this value is used, the PDCap1.1 byte holds the VCTID in a way that needs to be defined later.
		XX00 0001	MIFARE Plus Security Level 1/MIFARE Classic
		XX10 0001	MIFARE Plus Security Level 1 with sectors switched to higher security level
		XX00 0010	Obsolete MIFARE Plus Security Level 2
		XX00 0011	MIFARE Plus Security Level 3
		XX00 0100	MIFARE DESFire
		XX00 0101	MIFARE Ultralight
		XX00 0110	MIFARE Ultralight C
		XX00 0111	MIFARE Plus Security Level 0
		XX00 1000	RFU
		XX00 1001	MF3E(H)x3
		XX11 1111	Unsupported IID (after ISO14443-3 activation)
PDCap1.1		0000 0000	Reserved for future use. Chip maker Defined Capabilities. The interpretation of this byte depends on the VCTID. Different types of cards can communicate different or differently coded capabilities.

Provided under NDA only
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, distribute or reproduce
5fc07700-02c-40fc-bfc3cae42f1
Personal copy for:

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 50. PD capability information...*continued*

Capabilities Bytes	Information Field	Value	Description
PDCap1.2			Service Operator Defined Capability. This byte is to be configured then with SetConfiguration. It can be set by the service provider to whatever value is desired. It is intended to communicate those capabilities to the PCD where the different VCs belonging to the installation differ. Capabilities that are the same for all VCs of an installation (so VCs having the same VCIID) can be hard-coded in the PCD and do not need to be exchanged. This way of dealing with capabilities is chosen because of the limited size of the capability vector. The Service Operator is free to redefine the meaning of the PDCap1.2 byte during personalization/prepersonalization. The indicators below either need to be coded in this byte, or need to be known to the PCD in another way (hard coded because they are fixed for all VCs with this VCIID).
	CRDSIZ		Card Size: The amount of memory on the Virtual Card
		XXXX X000	<1 kB
		XXXX X001	1 kB
		XXXX X010	2 kB
		XXXX X011	4 kB
		XXXX X100	8 kB
		XXXX X101	16 kB
		XXXX X110	32 kB
		XXXX X111	Reserve for other sizes
	PCSUP		Proximity Check supported/mandatory
		XXX0 0XXX	Proximity Check not supported
		XXX0 1XXX	Proximity Check supported, not mandatory
		XXX1 0XXX	Proximity Check mandatory
		XXX1 1XXX	Reserved
	MKSID	mmmX XXX	Commander Keyset ID It identifies the set of master keys to use to derive the VC diversified keys from. The concept of commander Keysets is not defined by the architecture of MIFARE Plus and is up to the operator to define.

7.20.4.1.2 PCD capabilities

There are no PCD capabilities exchanged.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.20.4.1.3 Capability negotiation

The PD will publish its maximum capabilities. The PCD will adapt its transaction to make use of the maximum PD capabilities that are also supported by the PCD.

Depending on the PD configuration, it may allow the PCD to use still protocol features which are not according to its maximum capabilities. This allows non-upgraded PCDs to still communicate with these PDs in, for example, a transition phase.

In most cases, a PD can be configured to disallow these sub-maximum features. This can then be implemented as soon as all PCDs of an infrastructure have been upgraded.

Note: *The integrity of the PD capabilities is not ensured by the ISOSelectFile response. If required, they can be confirmed in secure channel afterward, reading them out with GetConfiguration.*

7.20.4.2 Capability exchange during symmetric authentication

The capability exchange in MF3E(H)x3 authentication is done whenever the authentication is initiated by a AuthenticateEV2First sent by the PCD (when initiated by AuthenticateEV2NonFirst, there are no capabilities exchanged). Note that only when the authentication was successful the correct exchange of capabilities can be relied upon.

7.20.4.2.1 PD capabilities

The PD capabilities consist of six bytes as defined in Table 51.

PDCap2.1	PDCap2.2	PDCap2.3	PDCap2.4	PDCap2.5	PDCap2.6

Table 51. PD capabilities during authentication

Capabilities Bytes	Value	Description
PDCap2.1	0x00	Chipmaker Defined Capability. Reserve for future use.
PDCap2.2	0x00..0x03	Chipmaker Defined Capability. TransactionTimer.
PDCap2.3	0x00	Chipmaker Defined Capability. Reserve for future use.
PDCap2.4	0x00	Chipmaker Defined Capability. Reserve for future use.
PDCap2.5	0x00	Service Operator Defined Capability. This is to be configured with Set Configuration.
PDCap2.6	0x00	Service Operator Defined Capability. This is to be configured with Set Configuration.

The PDCap2.5 and PDCap2.6 can be used by the operator to convey to the PCD information about the specifics of the PD or VC.

For example, if the operator has different data structures on different VCs that have the same VCIID, the PDCap bytes can be used to convey the actual data structure to the PCD.

The PDCap2.3 to PDCap2.4 bytes are reserved for future use. The PD shall set them to 0x00.

Later versions of MF3E(H)x3 can have additional capabilities.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.20.4.2.2 PCD capabilities

The PCD capabilities consist of 6 bytes as defined in Table 52.

PCDCap2.1	PCDCap2.2	PCDCap2.3	PCDCap2.4	PCDCap2.5	PCDCap2.6

PCD Capabilities bytes can be left out of the transmission. In that case the value of the byte is interpreted by the PD to be 0x00.

If PCDCap2.x is left out, all bytes PCDCap2.y where y > x must be left out as well.

All capabilities are Chipmaker Defined Capabilities.

Table 52. PCD capabilities during authentication

Capabilities Bytes	Value	Description
PCDCap2.1	0x00	Chipmaker Defined Capability. Reserve for future use. Shall be set to 0x00.
PCDCap2.2	0x00	Chipmaker Defined Capability. Reserve for future use. Shall be set to 0x00.
PCDCap2.3	0x00	Chipmaker Defined Capability. Reserve for future use. Shall be set to 0x00.
PCDCap2.4	0x00	Chipmaker Defined Capability. Reserve for future use. Shall be set to 0x00.
PCDCap2.5	0x00	Chipmaker Defined Capability. Reserve for future use. Shall be set to 0x00.
PCDCap2.6	0x00	Chipmaker Defined Capability. Reserve for future use. Shall be set to 0x00.

Note: *The PD shall accept any value of those bytes and potentially any bytes following those.*

7.20.4.2.3 The capability negotiation

For PDCap2.5 and PDCap2.6 there is only limited capability negotiation. The PD tells the PCD its capabilities and it is up to the PCD to take decisions based on that.

For the other capabilities, the negotiation process must be as follows.

The PCD first sends its capabilities to the PD. Assuming that the PD can interpret the capabilities, it will return its capabilities, but "maximized" to the capabilities of the PCD.

For every future capability, it is defined which is the lower value to which a capability can be negotiated down.

If the PD does not understand a capability, that is, a bit that is set to 1 which it does not understand, it must ignore it and return the bit set to 0 when it returns its capabilities. This is for example, the case when an old PD is confronted with a more modern PCD.

If the PD is confronted with a PCD that has capabilities that the PD regards too low, then the PD returns its minimum capability that would have been acceptable. The PD must then however not authenticate.

This way of negotiating means that a PCD will never be confronted with options from a PD that it does not understand. Of course this is only the case if the PCD puts the PCDCap to zeroes. So if the PCD is programmed strictly to interpret the returned PD capabilities and refuses to communicate when it finds a capability that it does not understand, that will not lead to interoperability problems.

7.20.4.2.4 Extensibility

The format of AuthenticateEV2NonFirst is extensible. The command has a length field after the command code, and it must be followed by the number of bytes that the length field indicates. However, the number of bytes is only limited by the amount that the length field can indicate. In practice, the maximum number of bytes allowed for the extension will be conditioned by the maximum message length imposed by the lower layers of the protocol handlers.

Provided under NDA only
COMPANY CONFIDENTIAL - personal copy for:
corinna.sala@
Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fc57179d-c2e2-4056-90fe-6ec3ca4a1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Every PD must accept these bytes, regardless of whether it understands those or not.

In the current version of the architecture, 0 bytes to 6 bytes can be sent to express capabilities. It can be that future versions would need even more information, for example, an extra random number of 16 bytes to be sent. In such a case the AUTHF command would send a length field of 22 bytes, 6 bytes of capabilities and 16 bytes of random number.

To guarantee operability, the PD must only check on the minimum length of the operands on the command, not on a maximum length. The extra bytes must be ignored.

The extensibility of the messages from the PD is reached by the PCD indicating in the capability bytes that it is capable of handling the higher-level feature that needs the message from the PD to be extended. Then the PCD can be prepared to receive another size response.

This means that the PD may in this case already be acting on the PCD capability before it knows that this capability information is not tampered with. This could be exploited for a denial of service attack (the PD sending something back that the PCD does not understand), but such attacks are possible in many other ways anyway and have to be accepted.

The extensibility of the PD can lead to a privacy concern, as an eavesdropper can see that the PD has a capability for which an extended response is needed. That may give indications about the device that can be exploited. This is a risk that cannot be avoided.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae421f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.21 Originality Check architecture

7.21.1 Dynamic Originality Check Procedure using ECC

During manufacturing, MF3E(H)x3 is trust-provisioned with an ECC-based key pair and related certificate to allow verification of the genuineness of the IC. The originality check is done by executing a card-unilateral authentication through a challenge-response protocol. For privacy reasons, the same key pair is shared across a batch of ICs. Because the protocol creates a trace that potentially cannot be repudiated, the key pair is shared by all ICs in one production batch to reduce the privacy implications. On top, the feature can be disabled by overwriting the key pair and certificate.

7.21.1.1 Originality key pair

The Originality Check key pair (Priv.Orig, Pub.Orig) is trust-provisioned with the following properties. For KeyPolicy and Read/WriteAccess encoding, refer to the ManageKeyPair API definition.

- Shared key pair per batch.
- Priv.Orig is stored as KeyID.ECCPrivateKey KeyNo 0x01 at the PICC level, holding with the following default configuration:
  - NIST P-256.
  - KeyPolicy: 0x0080, only allowing ECC-based Unilateral Authentication.
  - Write-access: 0x30. This is allowing replacement with ManageKeyPair after an authentication granting KeyID.PICCMasterKey access rights in CommMode.Full.
  - KUCLimit: disabled.
- Pub.Orig is trusted via the certificate Cert.Orig, as specified in Section 7.21.1.3. It can also be retrieved with ExportKey.

In the more generic description of the authentication (Priv.B, Pub.B), is used for the Originality Check key pair (Priv.Orig, Pub.Orig).

7.21.1.2 Originality certificate

MF3E(H)x3 holds a trust-provisioned file at the PICC level holding the Originality Check certificate that is formatted according to Section 7.10.1.1.

The file holding the certificate is a FileType.StandardData file of 384 bytes with the following properties:

- FileNo = 0x01; ISO File ID = 0xEF01.
- FileAR.Read =0xE; FileAR.Write =0x0; FileAR.ReadWrite =0x0; FileAR.Change =0x0.
- Secure Dynamic Messaging and mirroring is not supported for this file.
- Additional access rights are not supported for this file.
- CommMode.Full.

This file will hold the NXP Originality Certificate Cert.Orig which is described in Section 7.10.1.4.

7.21.1.3 Card-unilateral authentication

The authentication supported by Priv.Orig is outlined in Section 7.5.7. If the KeyID.ECCPrivateKey (KeyNo 0x01 at the PICC level) has an enabled KeyUsageCtrLimit, a special delay mechanism is implemented to reduce potential denial-of-service risk, see Section 7.9.1.2.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Salata
Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fc077bd-c22c-405c-90fe-6ec3cae42f1

7.22 Trust Provisioning

This chapter gives an overview of the E2E Trust Provisioning aspects of the MF3E(H)x3. This consists of the keys and related certificates that are injected into the MF3E(H)x3 during manufacturing. The chapter is constructed as a starting point, referring to the detailed definitions in other chapters and collecting the assumptions to be fulfilled by Trust Provisioning.

The following items are provisioned in the default configurations:

- Originality key pair: (Priv.Orig,Pub.Orig) related certificate Cert.Orig. This is a shared key pair and certificate per batch. See Section 7.22.1.
- PICC functionality key pair: (Priv.PICC, Pub.PICC) and related certificate Cert.PICC. This is unique per IC. See Section 7.22.2.
- The NXP Delegated Application management keys KeyID.NXPDAMKeys. See Section 7.22.3.

Customized product variants may allow for further provisioning, see Section 7.22.4.

7.22.1 Originality check key pair and certificate

For Originality Checking purposes (see Originality Check Architecture, MF3E(H)x3 holds a PICC-level key pair (Priv.Orig, Pub.Orig) and related certificate Cert.Orig. More precisely, the KeyID.ECCPrivateKey Priv.Orig is stored at KeyNo 0x01, as specified in Section 7.21.1.1, and the Cert.Orig is stored in FileNo 0x01, as specified in Section 7.21.1.2. The format and content of Cert.Orig itself is specified in Section 7.10.1.4.

7.22.2 PICC functionality key pair and certificate

MF3E(H)x3 holds a PICC-level key pair (Priv.PICC, Pub.PICC) and related certificate Cert.PICC. This can be used to authenticate individual devices for executing PICC-level functionality. More precisely, the KeyID.ECCPrivateKey.

Priv.PICC is stored at KeyNo 0x00, as specified in Section 7.9.4.1, and the Cert.PICC is stored in FileNo 0x00, as specified in Section 7.10.1.2. The format and content of Cert.PICC itself is specified in Section 7.10.1.4.

7.22.3 NXP Delegated Application Management keys

NXP Delegated Application Management keys (KeyID.NXPDAMKeys) are defined in Section 7.8.4.1, more specifically Table 32, and Section 7.8.4.1.3. These keys allow for application management by NXP, see Section 7.11.2.1.4, if enabled for the IC.

7.22.4 Commercial customization options

NXP provides commercial customization options for trust-provisioning. This allows for a customer-dedicated delivery configuration.

This may include the provisioning of a customer-specific KeyID.CARootKey. This allows to do the initial personalizaiton with the ECC-based mutual authentication. By this, the need for a secure environment can be removed, compared to when doing the initial personalization based on the default AES keys.

Additionally, also customer-specific AES keys, certificates and/or a customized file system and configuration can be provisioned. Reach out to your local sales representative for more information.

Provided under NDA only. COMPANY CONFIDENTIAL. Personal copy for: Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

7.23 Security

7.23.1 Introduction

NXP Semiconductors has gained comprehensive security experience from developing more than six generations of certified secure microcontrollers and other security certified products.

The large number of approved features and the significant enhancements over the different generations of certified secure products and secure microcontrollers are the foundation of the security concept that is implemented in the MF3E(H)x3 product. These mentioned design features related to security ensure to protect the integrity and confidentiality of user data and applications.

The unique security design is built on over one hundred dedicated security mechanisms which create a dense protection shield with redundancy and multiple layers. The security mechanisms provide a comprehensive response to the wide variety of known and expected security attacks. As attacks evolve over time, the distributed approach of the implemented security architecture allows for more proactive and continuous enhancements of the security mechanisms compared to alternative and less versatile approaches. This makes the underlying security architecture of MF3E(H)x3 a future-proof concept that´s built into the product, that effectively counters side channel and fault attacks as well as reverse engineering efforts.

The following sections describe a subset of the security features that are implemented on MF3E(H)x3.

7.23.2 Reset

The following types of resets and reset sources can be distinguished:

- normal application power-on reset, triggered by the on-chip power-on reset circuit
- internal reset, triggerable by
  - Dedicated software reset
  - On-chip security sensors
  - Electrical operating condition category
  - Internal physical attack category
  - Data integrity protection category

Two different reset severities can be distinguished in the MF3E(H)x3 hardware. The"normal" chip resets and the "security" resets.

7.23.3 Sensor Architecture

The following sensors are implemented on MF3E(H)x3:

- Electrical operation condition category
  - Low Frequency Sensor
  - High Frequency Sensor
  - Low Voltage Sensor
  - High Voltage Sensor
- Internal physical attack category
  - Low Temperature Sensor
  - High Temperature Sensor
  - Light Sensors
  - Glitch Sensors
  - Active Shielding

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

  - ISO/IEC 14443 Frequency Sensor
- Data integrity protection category
  - RAM Integrity Error
  - ROM Integrity Error
  - FLASH Integrity Error
  - internal Bus and Register Integrity Error

7.23.4 Scalable Security

MF3E(H)x3 implements an error counter. The error counter uses a dedicated memory area in the FLASH within a dedicated, protected memory area. The error counter is decremented for security critical errors.

The Scalable Security feature enables additional security countermeasures during operation based on the error counter values to avoid exploitation of repeated attacks. This results in a significant performance degradation in case of repeated security resets, if MF3E(H)x3 detects that it is under security attacks. From system design perspective this behavior has to be considered to avoid a non functional system due to timeouts by the PCD in case of activated Scalable Security features. Therefore timeouts should be defined with significant margins in case of additional security countermeasures are activated.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8 Commands

8.1 Introduction

This section contains the full command set of MF3E(H)x3.

For each command a figure and a table with the detailed command API is given. The table description of the commands is normative.

Remark: In the figures throughout Section 8, the plain representation of the command is always given. The secure messaging is defined in Section 7.5. In the table, it is indicated which communication mode needs to be applied, see Section 7.5.4.

8.1.1 Overview tables (sorted by name)

8.1.1.1 Native commands (sorted by name)

Table 53. Overview of native commands, sorted by name

Name	CmdCode	Description
AbortTransaction	0xA7	Aborts all previous write access to backup integrated files
AuthenticateEV2First	0x71	Authenticate with AES keys
AuthenticateEV2NonFirst	0x77	Authenticate with AES keys
ChangeFileSettings	0x5F	Changes the access parameters of an existing file
ChangeKey	0xC4	Changes the targeted key
ChangeKeyEV2	0xC6	Change any key stored on the PICC
ChangeKeySettings	0x54	Change the key settings on PICC or application level
ClearRecordFile	0xEB	Resets a LinearRecord or CyclicRecord file to empty state
CommitReaderID	0xC8	Commits a ReaderID for the ongoing transaction
CommitTransaction	0xC7	Commits all previous write access to backup integrated files
CreateApplication	0xCA	Creates new applications on the PICC
CreateBackupDataFile	0xCB	Creates a Backup Data File
CreateCyclicRecordFile	0xC0	Creates a Cyclic Record File
CreateDelegatedApplication	0xC9	Creates delegated applications on the PICC
CreateLinearRecordFile	0xC1	Creates a Linear Record File
CreateStdDataFile	0xCD	Creates a Standard Data File
CreateTransactionMACFile	0xCE	Creates a Transaction MAC File
CreateValueFile	0xCC	Creates a Value File
Credit	0x0C	Increases a value stored in a FileType.Value file
CryptoRequest	0x4C	Executes a cryptographic operation
CryptoRequest_ECCSign	0x4C	Executes an ECC signature generation
CryptoRequest_Echo	0x4C	Echos provided command data
Debit	0xDC	Decreases a value stored in a FileType.Value file

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantum State Solutions GmbH
Do not copy, share or reproduce
5f0779a6-220c-4056-906f-6acc5cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 53. Overview of native commands, sorted by name...*continued*

Name	CmdCode	Description
DeleteApplication	0xDA	Permanently deactivates applications on the PICC
DeleteFile	0xDF	Permanently deactivates a file within the file directory of the currently selected application
ExportKey	0x47	Export the public key value of a KeyID.CARootKey
FinalizeKeySet	0x57	Finalize a key set
Format	0xFC	Releases the PICC or application user memory
FreeMem	0x6E	Returns the free memory available on the card
GetApplicationIDs	0x6A	Returns the Application IDentifiers of all active applications on a PICC
GetCardUID	0x51	Returns the UID
GetConfiguration	0x65	Retrieve configuration
GetDFNames	0x6D	Returns the the Application IDentifiers together with a File ID and (optionally) a DF Name of all active applications with ISO/IEC 7816-4 support
GetDelegatedInfo	0x69	Returns the parameters associated with a delegated application
GetFileCounters	0xF6	Get the file related counters
GetFileIDs	0x6F	Returns the File IDentifiers of all active files within the currently selected application
GetFileSettings	0xF5	Get information on the properties of a specific file
GetISOFileIDs	0x61	Get back the ISO File IDs
GetKeySettings	0x45	Get key-related metadata
GetKeyVersion	0x64	Reads out the current version of any key or key set stored on the PICC
GetValue	0x6C	Reads the currently stored value from a FileType.Value file
GetVersion	0x60	Returns manufacturing related data of the PICC
InitializeKeySet	0x56	Initialize a key set
LimitedCredit	0x1C	Allows a limited increase of a value stored in a File-Type. Value file without having full Credit permissions to the file
ManageCARootKey	0x48	Manage a KeyID.CARootKey
ManageGPIO	0x42	Manages the GPIO output
ManageKeyPair	0x46	Manage a KeyID.ECCPrivateKey
ReadData	0xBD/0xAD	Reads data from FileType.StandardData, File-Type.BackupData or FileType.TransactionMAC files
ReadGPIO	0x43	Returns the GPIO status
ReadRecords	0xBB/0xAB	Reads out a set of complete records from a File-Type.LinearRecord or FileType.CyclicRecord File
RollKeySet	0x55	Roll to a new key set
SelectApplication	0x5A	Selects one or two specific applications or the PICC level for further access
SetConfiguration	0x5C	Configures the card or the application

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fd7f994-5202-40fc-90fc-6ec3ce2bf1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 53. Overview of native commands, sorted by name...*continued*

Name	CmdCode	Description
UpdateRecord	0xDB/0xBA	Updates data of an existing record in a File-Type.LinearRecord or File Type.CyclicRecord file
WriteData	0x3D/0x8D	Writes data to FileType.StandardData and File-Type.BackupData files
WriteRecord	0x3B/0x8B	Writes data to a record in a FileType.LinearRecord or File-Type.Cyclic Record file

8.1.1.1.1 Native command response codes (sorted by name)

Table 54. Overview of native response codes

Name	RespCode	Description
Resp.ADDITIONAL_FRAME	0xAF	Additional data frame is expected to be sent
Resp.APPLICATION_NOT_FOUND	0xA0	Requested AID not present on PICC
Resp.AUTHENTICATION_ERROR	0xAE	Current authentication status does not allow the requested command
Resp.BOUNDARY_ERROR	0xBE	Attempt to read/write data from/to beyond the file's/record'slimits. Attempt to exceed the limits of avalue file
Resp.CERT_ERROR	0xCE	Readercertificate or CertAccessRights related error
Resp.COMMAND_ABORTED	0xCA	Previous command was not fully completed. Not all Frames were requested or provided by the PCD
Resp.DUPLICATE_ERROR	0xDE	Creatio nof file/application failed because file/application withsame number already exists
Resp.FILE_NOT_FOUND	0xF0	Specified file number does not exist
Resp.ILLEGAL_COMMAND_CODE	0x1C	Command code not supported
Resp.INTEGRITY_ERROR	0x1E	CRCor MAC does not match data. Padding bytes not valid
Resp.LENGTH_ERROR	0x7E	Length of command string invalid
Resp.NO_CHANGES	0x0C	No changes done to backup files, CommitTransaction /Abort Transaction not necessary
Resp.NO_SUCH_KEY	0x40	Invalidkey number specified
Resp.OPERATION_OK	0x00	Successful operation
Resp.OPERATION_OK_LIM	0x01	Successful operation, but limited functionality
Resp.OUT_OF_MEMORY_ERROR	0x0E	InsufficientNV-Memory to complete command
Resp.PARAMETER_ERROR	0x9E	Value of the parameter(s) invalid
Resp.PERMISSION_DENIED	0x9D	Current configuration / status does not allow the requested command
Resp.WEAK_FIELD	0x1F	Field strength not sufficient to enable power downstream for the targeted current/voltage level

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.1.1.2 PC commands (sorted by name)

Table 55. Overview PC related commands, sorted by name

Name	CmdCode	Description
PreparePC	0xF0	Prepare for the Proximity Check
ProximityCheck	0xF2	Perform the precise measurement for the Proximity Check
VerifyPC	0xFD	Verify the Proximity Check

8.1.1.2.1 PC command response codes (sorted by name)

Table 56. Overview PC related response codes

Name	RespCode	Description
Resp.COMMAND_ABORTED	0xCA	Previous Command was not fully completed. Not all Frames were requested or provided by the PCD
Resp.ErrAuth	0x06	Authentication failure
Resp.ErrBNr	0x09	InvalidBlock number: not existing in the implementation or not valid to target with this command
Resp.ErrCmdInvalid	0x0B	Command is received in a state where this command is not supported, or a totally unknown command is received
Resp.ErrCmdOverflow	0x07	Too many commands in the session or transaction
Resp.ErrFormat	0x0C	Format of the command is not correct (e.g. too many or too few bytes)
Resp.ErrGenFailure	0x0F	Failure in the operation of the PD
Resp.OK	0x90	Success

8.1.1.3 ISO7816-4 standard commands (sorted by name)

Table 57. Overview of ISO7816-4 commands

Name	CmdCode	Description
ISOAppendRecord	0xE2	Writea new record to a record file
ISOGeneralAuthenticate	0x87	Asymmetric mutual or reader-unilateral authentication.
ISOGeneralAuthenticateFinal	0x87	Asymmetric mutual or reader-unilateral authentication.
ISOGeneralAuthenticatePart2	0x87	Asymmetric mutual or reader-unilateral authentication.
ISOGetChallenge	0x84	Get a challenge (first part of ISO7816-4 authentication)
ISOInternalAuthenticate	0x88	Asymmetric card-unilateral authentication.
ISOReadBinary	0xB0	Read from a data file
ISOReadRecord	0xB2	Read from a record file
ISOSelectFile	0xA4	Select an application or file
ISOUpdateBinary	0xD6	Write to a data file
VDE_ECDSASign	0x03	Executes an ECC signature generation according to VDE-AR-E2532-100
VDE_ReadData	0x02	Reads data according to VDE-AR-E 2532-100
VDE_WriteData	0x01	Writes data to FileNo 0x01 according to VDE-AR-E 2532-100

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.1.1.3.1 ISO7846-4 standard command response codes (sorted by name)

Table 58. Overview of ISO7816-4 response codes (SW1|SW2)

Name	RespCode	Description
Resp.ISO6283	0x6283	Selected file or application deactivated: selected with limited functionality
Resp.ISO6581	0x6581	Memory failure
Resp.ISO6700	0x6700	Wrong length; no further indication
Resp.ISO6982	0x6982	Security status not satisfied
Resp.ISO6984	0x6984	Reference data not usable
Resp.ISO6985	0x6985	Conditions of use not satisfied
Resp.ISO6987	0x6987	Expected secure messaging DOs missing
Resp.ISO6988	0x6988	Incorrect secure messaging DOs
Resp.ISO6A80	0x6A80	Incorrect parameters in the command data field
Resp.ISO6A82	0x6A82	File or application not found
Resp.ISO6A86	0x6A86	Incorrect parameters P1-P2
Resp.ISO6A87	0x6A87	Lc inconsistent with parameters P1-P2
Resp.ISO6A88	0x6A88	Referenced data, reference data or DO not found
Resp.ISO6C00	0x6C00	Wrong Le field
Resp.ISO6D00	0x6D00	Instruction code not supported or invalid
Resp.ISO9000	0x9000	Normal processing (no further qualification)
Resp.ISO9FXX	0x9FXX	Correct execution. Requested protocol not supported

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.1.2 Overview tables (sorted by respond/command code)

8.1.2.1 Native commands (sorted by command code)

Table 59. Overview of native commands, sorted by command code

CmdCode	Name	Description
0x0C	Credit	Increases a value stored in a FileType.Value file
0x1C	LimitedCredit	Allows a limited increase of a value stored in a FileType. Value file without having full Credit permissions to the file
0x3B/0x8B	WriteRecord	Writes data to a record in a FileType.LinearRecord or File-Type.CyclicRecord file
0x3D/0x8D	WriteData	Writes data to FileType.StandardData and File-Type.BackupData files
0x42	ManageGPIO	Manages the GPIO output
0x43	ReadGPIO	Returns the GPIO statuses
0x45	GetKeySettings	Get key-related metadata
0x51	GetCardUID	Returns the UID
0x46	ManageKeyPair	Manage a KeyID.ECCPrivateKey
0x47	ExportKey	Export the public key value of a KeyID.CARootKey
0x48	ManageCARootKey	Manage a KeyID.CARootKey
0x4C	CryptoRequest	Executes a cryptographic operation
0x4C	CryptoRequest_ECCSign	Executes an ECC signature generation
0x4C	CryptoRequest_Echo	Echos provided command data
0x54	ChangeKeySettings	Change the key settings on PICC or application level
0x55	RollKeySet	Roll to a new key set
0x56	InitializeKeySet	Initialize a key set
0x57	FinalizeKeySet	Finalize a key set
0x5A	SelectApplication	Selects one or two specific applications or the PICC level for further access
0x5C	SetConfiguration	Configures the card or the application
0x5F	ChangeFileSettings	Changes the access parameters of an existing file
0x60	GetVersion	Returns manufacturing related data of the PICC
0x61	GetISOFileIDs	Get back the ISO File IDs
0x64	GetKeyVersion	Reads out the current version of any key or key set stored on the PICC
0x65	GetConfiguration	Retrieve configuration
0x69	GetDelegatedInfo	Returns the parameters associated with a delegated application
0x6A	GetApplicationIDs	Returns the Application IDentifiers of all active applications on a PICC
0x6C	GetValue	Reads the currently stored value from a FileType.Valuefile
0x6D	GetDFNames	Returns the the Application IDentifiers together with a File ID and (optionally) a DF Name of all active applications withISO/IEC 7816-4 support
0x6E	FreeMem	Returns the free memory available on the card

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 59. Overview of native commands, sorted by command code...*continued*

CmdCode	Name	Description
0x6F	GetFileIDs	Returns the File IDentifiers of all active files within the currently selected application
0x71	AuthenticateEV2First	Authenticate with AES keys
0x77	AuthenticateEV2NonFirst	Authenticate with AES keys
0xA7	AbortTransaction	Aborts all previous write access to backup integrated files
0xAB	ReadRecords	Reads out a set of complete records from a FileType.LinearRecord or File Type.CyclicRecord File
0xAD	ReadData	Reads data from FileType.StandardData, FileType.BackupData or FileType. TransactionMAC files
0xBA	UpdateRecord	Updates data of an existing record in a FileType.LinearRecord or FileType. CyclicRecord file
0xBB	ReadRecords	Reads out a set of complete records from a FileType.LinearRecord or File Type.CyclicRecord File
0xBD	ReadData	Reads data from FileType.StandardData, FileType.BackupData or FileType. TransactionMAC files
0xC0	CreateCyclicRecordFile	Creates a Cyclic Record File
0xC1	CreateLinearRecordFile	Creates a Linear Record File
0xC4	ChangeKey	Changes the targeted key
0xC6	ChangeKeyEV2	Change any key stored on the PICC
0xC7	CommitTransaction	Commits all previous write access to backup integrated files
0xC8	CommitReaderID	Commits a ReaderID for the ongoing transaction
0xC9	CreateDelegated Application	Creates delegated applications on the PICC
0xCA	CreateApplication	Creates new applications on the PICC
0xCB	CreateBackupDataFile	Creates a Backup Data File
0xCC	CreateValueFile	Creates a Value File
0xCD	CreateStdDataFile	Creates a Standard Data File
0xCE	CreateTransactionMACFile	Creates a Transaction MAC File
0xDA	DeleteApplication	Permanently deactivates applications on the PICC
0xDB	UpdateRecord	Updates data of an existing record in a FileType.LinearRecord or FileType. CyclicRecord file
0xDC	Debit	Decreases a value stored in a FileType.Value file
0xDF	DeleteFile	Permanently deactivates a file within the file directory of the currently selected application
0xEB	ClearRecordFile	Resets a FileType.LinearRecord or FileType.CyclicRecord file to empty state
0xF5	GetFileSettings	Get information on the properties of a specific file
0xF6	GetFileCounters	Get the file related counters
0xFC	Format	Releases the PICC or application user memory

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.1.2.1.1 Native Command Response Codes (sorted by code)

Table 60. Overview of native response codes, sorted by response code

RespCode	Name	Description
0x00	Resp.OPERATION_OK	Successful operation
0x01	Resp.OPERATION_OK_LIM	Successful operation, but limited functionality
0x0C	Resp.NO_CHANGES	No changes done to backup files, CommitTransaction / Abort Transaction not necessary
0x0E	Resp.OUT_OF_MEMORY_ERROR	Insufficient NV-Memory to complete command
0x1C	Resp.ILLEGAL_COMMAND_CODE	Command code not supported
0x1E	Resp.INTEGRITY_ERROR	CRCor MAC does not match data. Padding bytes not valid
0x1F	Resp.WEAK_FIELD	Field strength not sufficient to enable power downstream for the targeted current/voltage level
0x40	Resp.NO_SUCH_KEY	Invalid key number specified
0x7E	Resp.LENGTH_ERROR	Length of command string invalid
0x9D	Resp.PERMISSION_DENIED	Current configuration / status does not allow the requested command
0x9E	Resp.PARAMETER_ERROR	Value of the parameter(s) invalid
0xA0	Resp.APPLICATION_NOT_FOUND	Requested AID not present on PICC
0xAE	Resp.AUTHENTICATION_ERROR	Current authentication status does not allow the requested command
0xAF	Resp.ADDITIONAL_FRAME	Additional data frame is expected to be sent
0xBE	Resp.BOUNDARY_ERROR	Attempt to read/write data from/to beyond the file's/record's limits. Attempt to exceed the limits of a value file
0xCA	Resp.COMMAND_ABORTED	Previous Command was not fully completed. Not all Frames were requested or provided by the PCD
0xCE	Resp.CERT_ERROR	Reader certificate or CertAccessRights related error
0xDE	Resp.DUPLICATE_ERROR	Creation of file/application failed because file/application with same number already exists
0xF0	Resp.FILE_NOT_FOUND	Specified file number does not exist

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-4035-9ate (ec3ca42f)

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.1.2.2 PC commands (sorted by command code)

Table 61. Overview PC related commands, sorted by command code

CmdCode	Name	Description
0xF0	PreparePC	Prepare for the Proximity Check
0xF2	ProximityCheck	Perform the precise measurement for the Proximity Check
0xFD	VerifyPC	Verify the Proximity Check

8.1.2.2.1 PC Comand Response Codes (sorted by code)

Table 62. Overview PC related response codes (sorted by response code)

RespCode	Name	Description
0x06	Resp.ErrAuth	Authentication failure
0x07	Resp.ErrCmdOverflow	Too many commands in the session or transaction
0x09	Resp.ErrBNr	InvalidBlock number: not existing in the implementation or not valid to target with this command
0x0B	Resp.ErrCmdInvalid	Command is received in a state where this command is not supported, or a totally unknown command is received
0x0C	Resp.ErrFormat	Format of the command is not correct (e.g. too many or too few bytes)
0x0F	Resp.ErrGenFailure	Failure in the operation of the PD
0x90	Resp.OK	Success
0xCA	Resp.COMMAND_ABORTED	Previous Command was not fully completed. Not all Frames were requested or provided by the PCD

8.1.2.3 ISO7816-4 standard commands (sorted by command code)

Table 63. Overview of ISO7816-4 commands, sorted by command code (INS)

CmdCode	Name	Description
0x01	VDE_WriteData	Writes data to FileNo 0x01 according to VDE-AR-E 2532-100
0x02	VDE_ReadData	Reads data according to VDE-AR-E 2532-100
0x03	VDE_ECDSASign	Executes an ECC signature generation according to VDE-AR-E2532-100
0x84	ISOGetChallenge	Get a challenge (first part of ISO7816-4 authentication)
0x87	ISOGeneralAuthenticate	Asymmetric mutual or reader-unilateral authentication
0x87	ISOGeneralAuthenticateFinal	Asymmetric mutual or reader-unilateral authentication
0x87	ISOGeneralAuthenticatePart2	Asymmetric mutual or reader-unilateral authentication
0x88	ISOInternalAuthenticate	Asymmetric card-unilateral authentication
0xA4	ISOSelectFile	Select an application or file
0xB0	ISOReadBinary	Read from a data file
0xB2	ISOReadRecord	Read from a record file
0xD6	ISOUpdateBinary	Write to a data file
0xE2	ISOAppendRecord	Write a new record to a record file

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.1.2.3.1 ISO7846-4 Standard Command Response Codes (sorted by code)

Table 64. Overview of ISO7816-4 response codes (SW1|SW2) (sorted by response code)

RespCode	Name	Description
0x6283	Resp.ISO6283	Selected file or application deactivated: selected with limited functionality
0x6581	Resp.ISO6581	Memory failure
0x6700	Resp.ISO6700	Wrong length; no further indication
0x6982	Resp.ISO6982	Security status not satisfied
0x6984	Resp.ISO6984	Reference data not usable
0x6985	Resp.ISO6985	Conditions of use not satisfied
0x6987	Resp.ISO6987	Expected secure messaging DOs missing
0x6988	Resp.ISO6988	Incorrect secure messaging DOs
0x6A80	Resp.ISO6A80	Incorrect parameters in the command data field
0x6A82	Resp.ISO6A82	File or application not found
0x6A86	Resp.ISO6A86	Incorrect parameters P1-P2
0x6A87	Resp.ISO6A87	Lc inconsistent with parameters P1-P2
0x6A88	Resp.ISO6A88	Referenced data, reference data or DO not found
0x6C00	Resp.ISO6C00	Wrong Le field
0x6D00	Resp.ISO6D00	Instruction code not supported or invalid
0x9000	Resp.ISO9000	Normal processing (no further qualification)
0x9FXX	Resp.ISO9FXX	Correct execution. Requested protocol not supported.

Provided under NDA only
COMPANY CONFIDENTIAL - personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc5cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.2 Authentication and secure messaging

8.2.1 ISOGeneralAuthenticate

The detailed description of this command can be found in Section 7.5.6.5.

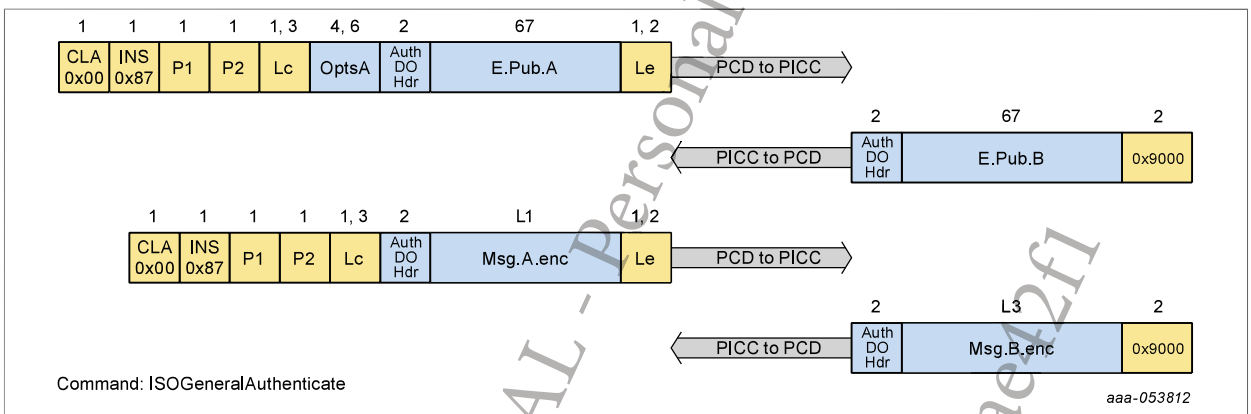


Figure 25. ISOGeneralAuthenticate command protocol

Table 65. Command description - ISOGeneralAuthenticate

ISOGeneralAuthenticate	
Description:	Asymmetric mutual or reader-unilateral authentication. This is the first part initiating the authentication.

Table 66. Command parameters description - ISOGeneralAuthenticate

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0x87	
P1	1	0x00	RFU
P2	1	-	Key and certificate addressing
	Bit 7		RFU
		0b	
	Bit 6-4		secCARootKeyNo
		Limited range	[if multiple application selection] Key number of the secondary targeted CA Root Key. Depending on available keys.
		000b	[else] RFU
	Bit 3		RFU
		0b	
	Bit 2-0		CARootKeyNo
		Limited range	Key number of the (primary) targeted CA Root Key. Depending on available keys.

Provided under NDA on behalf of: COMPANY CONFIDENTIAL - Personal copy for: Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc779d-c22-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 66. Command parameters description - ISOGeneralAuthenticate...*continued*

Name	Length	Value	Description
Lc	1,3	0x49..0xFFFF	Length of subsequent data field
OptsA	4, 6	-	PCD Option (TLV)
		T: 0x80	Tag
		L: 0x02, 0x04	Length of Value field
		V[0]: 0x80	Value: AuthMethod asymmetric mutual authentication with Cert.A
		V[0]: 0xA0	Value: AuthMethod asymmetric mutual authentication without Cert.A
		V[0]: 0x40	Value: AuthMethod asymmetric reader-unilateral authentication with Cert.A
		V[0]: 0x60	Value: AuthMethod asymmetric reader-unilateral authentication without Cert.A
		V[1]: 0x00	Value: ProtocolVersion
		V[2]: CertFileNo	[Present only if V[0].Bit7-6 =='10'] Value: Certificate File Number Bit 7: SAI Bit 6-5: RFU Bit 4-0: File number
		V[3]: PrivKeyNo	[Present only if V[0].Bit7-6 =='10'] Value: Private Key Number Bit 7-3: RFU Bit 2-0: Key number
AuthDOHdr	2	-	Authentication Data Objects Header (TL)
		T: 0x7C	Tag
		L: 0x43	Length of subsequent Authentication Data Objects
E.Pub.A	67	-	Authentication Data Object: ephemeral public key from PCD (TLV)
		T: 0x85	Tag
		L: 0x41	Length of Value field
		V: E.Pub.A	Value: ephemeral public key in uncompressed point representation (0x04 || E.Pub.A.x || E.Pub.A.y)
Le	1,2	-	Length of expected response
		0x00/0x0000	Any expected length up to resp. 256/65536 bytes.
		0x45 .. 0xFFFF	Max expected length must be at least 69 bytes.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Sadata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-495c-90fe-6ecc3cae4ff1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 67. Return code description - ISOGeneralAuthenticate

Status	Value	Description
ISO9000	0x9000	Correct execution
ISO9FXX	0x9FXX	[XX == 0x00] Correct execution. Requested ProtocolVersion not supported. Fallback to ProtocolVersion 0x00. See ISO9000 for response data.
ISO6700	0x6700	Wrong or inconsistent APDU length.
ISO6982	0x6982	Security status not satisfied: public key validation for E.Pub.A failed.
ISO6984	0x6984	Secondary KeyID.CARootKey targeted by secCARootKeyNo in case of multiple application selection has a different public key value, curve ID or issuer name as the primary KeyID.CARootKey targeted by CARootKeyNo.
		Secondary KeyID.CARootKey targeted by secCARootKeyNo in case of multiple application selection has a CRLFile associated while the primary KeyID.CARootKey targeted by CARootKeyNo has not, or vice versa.
		Secondary KeyID.CARootKey targeted by secCARootKeyNo in case of multiple application selection has a different CRLFile associated than the primary KeyID.CARootKey targeted by CARootKeyNo.
		File targeted by CertFileNo does not exist.
		SAI for CertFileNo given but no 2nd application selected.
		Key targeted by PrivKeyNo does not exist.
		Key targeted by PrivKeyNo does not have ECC-based Muatual Authentication enabled in its key policy.
		Key targeted by PrivKeyNo has enabled KeyUsageCtrLimit reached.
ISO6985	0x6985	Wrapped chained command or multiple pass command ongoing.
		ECC-based Mutual and Reader-Unilateral Authentication disabled over NFC interface.
		ECC-based Mutual and Reader-Unilateral Authentication disabled over I^2C interface.
		Targeted CA Root Key and private key belong to different curve.
ISO6987	0x6987	Expected DO missing.
ISO6988	0x6988	Unexpected DO received.
ISO6A86	0x6A86	Wrong parameter P1: different from 0x00.
		Wrong parameter P2: RFU bits set.
ISO6A88	0x6A88	Wrong parameter P2: CA Root Key targeted by CARootKeyNo does not exist.
		Wrong parameter P2: CA Root Key targeted by secCARootKeyNo does not exist in secondary application, or is different from '000' while only single application is selected.
ISO6C00	0x6C00	Wrong Le: expected length insufficient for response data.

Table 68. Response data parameters description - ISOGeneralAuthenticate - ISO9000

Name	Length	Value	Description
AuthDOHdr	2	-	Authentication Data Objects Header (TL)
		T: 0x7C	Tag
		L: 0x43	Length of subsequent Authentication Data Objects

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 68. Response data parameters description - ISOGeneralAuthenticate - ISO9000*...continued*

Name	Length	Value	Description
E.Pub.B	67	-	Authentication Data Objects: ephemeral public key from PICC (TLV)
		T: 0x85	Tag
		L: 0x41	Length of Value field
		V: E.Pub.B	Value: ephemeral public key in uncompressed point representation (0x04 || E.Pub.B.x || E.Pub.B.y)

Table 69. Command description - ISOGeneralAuthenticatePart2

ISOGeneralAuthenticatePart2	
Description:	Asymmetric mutual or reader-unilateral authentication. This is the second part concluding the authentication initiated by ISOGeneralAuthenticate .

Table 70. Command parameters description - ISOGeneralAuthenticatePart2

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0x87	
P1	1	0x00	RFU
P2	1	0x00	RFU
Lc	1,3	L1+2	Length of subsequent data field
AuthDOHdr	2	-	Authentication Data Objects Header (TL)
		T: 0x7C	Tag
		L: L1	Length of subsequent Authentication Data Objects
Msg.A.enc	L1	-	Authentication Data Object: Enciphered data from PCD (TLV)
		T: 0x86	Tag
		L: L2	Length of Value field
		V: Msg.A.enc	Value: Enciphered signature and optional certificate (chain): E($K_{SesAuthENCKey}$, 0xE0E0 [|| Cert.A] || Sig.A)
Le	1,2	-	Length of expected response
		0x00	Any expected length up to 256 bytes: rejected if (L3 + 2) > 256
		0x0000	Any expected length up to 65536 bytes.
		0xXX/0xXXXX	Max expected length: must be bigger than (L3 + 2)

Table 71. Return code description - ISOGeneralAuthenticatePart2

Status	Value	Description
ISO9000	0x9000	Correct execution
ISO6700	0x6700	Wrong or inconsistent APDU length.
ISO6982	0x6982	Security status not satisfied: Msg.A.enc invalid padding.

Provided under NDA only
COMPANY CONFIDENTIAL - personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 71. Return code description - ISOGeneralAuthenticatePart2*...continued*

Status	Value	Description
		Security status not satisfied: Sig.A verification failed.
		Security status not satisfied: Cert.A signature validation failed.
		Security status not satisfied: Cert.A.leaf signature validation failed.
		Security status not satisfied: Cert.A.parent signature validation failed.
		Security status not satisfied: Cert.A revoked.
		Security status not satisfied: Cert.A.leaf revoked.
		Security status not satisfied: Cert.A.parent revoked.
		Security status not satisfied: FileAR.Read or FileAR.ReadWrite not granted to the file targeted by CertFileNo.
ISO6985	0x6985	Wrapped chained command or multiple pass command ongoing.
		File targeted by CertFileNo is not a FileType.StandardData or FileType.BackupData .
ISO6984	0x6984	File targeted by CertFileNo has invalid content: CertificateLength > Filesize - 3.
ISO6987	0x6987	Expected DO missing.
ISO6988	0x6988	Unexpected DO received.
ISO6A80	0x6A80	Incorrect parameter: Cert.A omitted while expected or vice versa.
		Incorrect parameter: Cert.A signature format or chaining validation failed.
		Incorrect parameter: Cert.A.leaf signature format or chaining validation failed.
		Incorrect parameter: Cert.A.parent signature format or chaining validation failed.
ISO6A86	0x6A86	Wrong parameter P1: different from 0x00.
		Wrong parameter P2: different from 0x00.
ISO6C00	0x6C00	Wrong Le: expected length insufficient for response data.

Provided under NDA only. Personal copy for: COMPANY CONFIDENTIAL Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-405c-90fe-6ec5cae40f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 72. Response data parameters description - ISOGeneralAuthenticatePart2 - ISO9000

Name	Length	Value	Description
AuthDOHdr	2	-	Authentication Data Objects Header (TL)
		T: 0x7C	Tag
		L: L3	Length of subsequent Authentication Data Objects
Msg.B.enc	L3	-	Authentication Data Objects: Enciphered response from PICC (TLV)
		T: 0x82	Tag
		L: L4	Length of Value field
		V: Msg.B.enc	[if asymmetric mutual authentication] Value: Enciphered response: $E(K_{SesAuthENCKey}, 0xE1E1 \| \| OptsB \| \| Cert.B \| \| Sig.B)$
		V: Msg.B.enc	[if asymmetric reader-unilateral authentication] Value: Enciphered response: $E(K_{SesAuthENCKey}, 0xE1E1 \| \| OptsB)$

Provided under NDA only
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
Personal copy for:
5fc0779d-c22c-405c-90fe-6ecc3cae4211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.2.2 ISOGeneralAuthenticateFinal

The detailed description of this command can be found in Section 7.5.6.6.

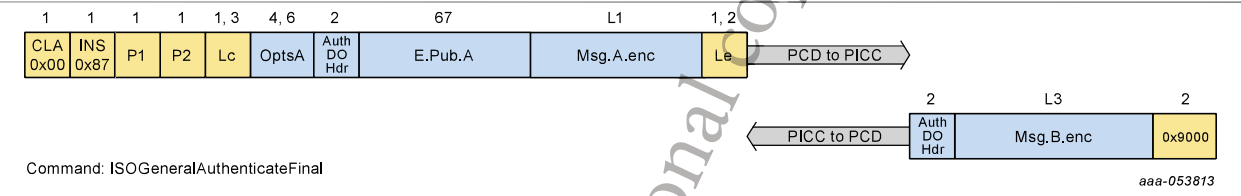


Figure 26. ISOGeneralAuthenticateFinal command protocol

Table 73. Command description - ISOGeneralAuthenticateFinal

ISOGeneralAuthenticateFinal	
Description:	Asymmetric mutual or reader-unilateral authentication. This is the second part following ISOSelectFile .

Table 74. Command parameters description - ISOGeneralAuthenticateFinal

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0x87	
P1	1	0x00	RFU
P2	1	-	KeyRefs
	Bit 7-3		RFU
		00000b	
	Bit 2-0		CARootKeyNo
		Limited range	Key number of the targeted CA Root Key. Depending on avail- able keys.
Lc	1,3	L1+72 .. 0xFFFF	Length of subsequent data field
OptsA	4, 6	-	PCD Option (TLV)
		T: 0x80	Tag
		L: 0x02, 0x04	Length of Value field
		V[0]: 0x80	Value: AuthMethod asymmetric mutual authentication with Cert.A
		V[0]: 0xA0	Value: AuthMethod asymmetric mutual authentication without Cert.A
		V[0]: 0x40	Value: AuthMethod asymmetric reader-unilateral authentication with Cert.A
		V[0]: 0x60	Value: AuthMethod asymmetric reader-unilateral authentication without Cert.A
		V[1]: 0x00	Value: ProtocolVersion
		V[2]: CertFileNo	[Present only if V[0].Bit7-6 =='10'] Value: Certificate File Number Bit 7-5: RFU Bit 4-0: File number
		V[3]: PrivKeyNo	[Present only if V[0].Bit7-6 =='10'] Value: Private Key Number Bit 7-3: RFU Bit 2-0: Key number
AuthDOHdr	2	-	Authentication Data Objects Header (TL)
		T: 0x7C	Tag

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07794-c22c-405c-90fe-6ecc3cae4211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 74. Command parameters description - ISOGeneralAuthenticateFinal...*continued*

Name	Length	Value	Description
		L: 67+ L1	Length of subsequent Authentication Data Objects
E.Pub.A	67	-	Authentication Data Object: ephemeral public key from PCD (TLV)
		T: 0x85	Tag
		L: 0x41	Length of Value field
		V: E.Pub.A	Value: ephemeral public key in uncompressed point representation (0x04 || E.Pub.A.x || E.Pub.A.y)
Msg.A.enc	L1	-	Authentication Data Object: Enciphered data from PCD (TLV)
		T: 0x86	Tag
		L: L2	Length of Value field
		V: Msg.A.enc	Value: Enciphered signature and certificate (chain): E($K_{SesAuthENCKey}$, 0xE0E0 [|| Cert.A] || Sig.A)
Le	1,2	-	Length of expected response
		0x00	Any expected length, rejected if (L3 + 2) > 256
		0x0000	Any expected length
		0xXX/0xXXXX	Max expected length: must be bigger than (L3 + 2)

Table 75. Return code description - ISOGeneralAuthenticateFinal

Status	Value	Description
ISO9000	0x9000	Correct execution
ISO9FXX	0x9FXX	[XX == 0x00] Correct execution. Requested ProtocolVersion not supported. Fallback to ProtocolVersion 0x00. See ISO9000 for response data.
ISO6700	0x6700	Wrong or inconsistent APDU length.
ISO6982	0x6982	Security status not satisfied: public key validation for E.Pub.A failed.
		Security status not satisfied: Msg.A.enc invalid padding.
		Security status not satisfied: Sig.A verification failed.
		Security status not satisfied: Cert.A signature validation failed.
		Security status not satisfied: Cert.A.leaf signature validation failed.
		Security status not satisfied: Cert.A.parent signature validation failed.
		Security status not satisfied: Cert.A revoked.
		Security status not satisfied: Cert.A.leaf revoked.
		Security status not satisfied: Cert.A.parent revoked.
		Security status not satisfied: FileAR.Read or FileAR.ReadWrite not granted to the file targeted by CertFileNo.
ISO6984	0x6984	File targeted by CertFileNo does not exist.
		SAI for CertFileNo given but no 2nd application selected.
		Key targeted by PrivKeyNo does not exist.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07791-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 75. Return code description - ISOGeneralAuthenticateFinal...*continued*

Status	Value	Description
		Key targeted by PrivKeyNo does not have ECC-based Muatual Authentication enabled in its key policy.
		Key targeted by PrivKeyNo has enabled KeyUsageCtrLimit reached.
		File targeted by CertFileNo has invalid content: CertificateLength > Filesize - 3.
ISO6985	0x6985	Wrapped chained command or multiple pass command ongoing.
		File targeted by CertFileNo is not a FileType.StandardData or FileType.BackupData .
		Targeted CA Root Key does not belong to configured curve.
		Targeted private key does not belong to configured curve.
ISO6987	0x6987	Expected DO missing.
ISO6988	0x6988	Unexpected DO received.
ISO6A80	0x6A80	Incorrect parameter: Cert.A omitted while expected or vice versa.
		Incorrect parameter: Cert.A signature format or chaining validation failed.
		Incorrect parameter: Cert.A.leaf signature format or chaining validation failed.
		Incorrect parameter: Cert.A.parent signature format or chaining validation failed.
ISO6A86	0x6A86	Wrong parameter P1: different from 0x00.
		Wrong parameter P2: RFU bits set.
ISO6A88	0x6A88	Wrong parameter P2: CA Root Key targeted by CARootKeyNo does not exist.
ISO6C00	0x6C00	Wrong Le: expected length insufficient for response data.

Table 76. Response data parameters description - ISOGeneralAuthenticateFinal - ISO9000

Name	Length	Value	Description
AuthDOHdr	2	-	Authentication Data Objects Header (TL)
		T: 0x7C	Tag
		L: L3	Length of subsequent Authentication Data Objects
Msg.B.enc	L3	-	Authentication Data Objects: Enciphered response from PICC (TLV)
		T: 0x82	Tag
		L: L4	Length of Value field
		V: Msg.B.enc	[if asymmetric mutual authentication Value: Enciphered response: $E(K_{SesAuthENCKey}, 0xE1E1 \| OptsB \| Cert.B \| Sig.B)$
		V: Msg.B.enc	[if asymmetric reader-unilateral authentication] Value: Enciphered response: $E(K_{SesAuthENCKey}, 0xE1E1 \| OptsB)$

Provided under NDA only - Personal copy for:
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-2220-40ec-90fe-6ec3ca42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.2.3 ISOInternalAuthenticate

The detailed description of this command can be found in Section 7.5.7.4.

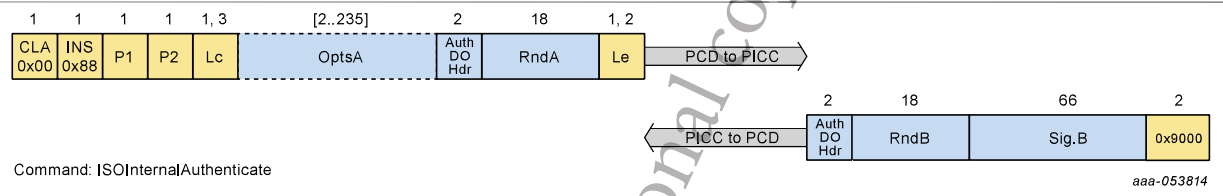


Figure 27. ISOInternalAuthenticate command protocol

Table 77. Command description - ISOInternalAuthenticate

ISOInternalAuthenticate	
Description:	Asymmetric card-unilateral authentication.

Table 78. Command parameters description - ISOInternalAuthenticate

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0x88	
P1	1	0x00	RFU
P2	1	-	Key addressing
	Bit 7-3		RFU
		00000b	
	Bit 2-0		PrivKeyNo
		0x0..0x1	[if PICC level (AID 0x000000) is selected] At PICC level, two keys are supported.
		0x0..0x4	[if PICC level is not selected] At application level, up to five keys are supported.
Lc	1,3	0x14..0xFF	Length of subsequent data field
OptsA	[2..235]	-	PCD Option (TLV): Optional (includes all Tag-Length-Value fields)
		T: 0x80	Tag
		L: 0x00..0xE8	Length of Value field. Card will accept other lengths and ignore the Value field.
AuthDOHdr	2	-	Authentication Data Objects Header (TL)
		T: 0x7C	Tag
		L: 0x12	Length of subsequent Authentication Data Objects
RndA	18	-	Authentication Data Object: random challenge from PCD (TLV)
		T: 0x81	Tag
		L: 0x10	Length of Value field
		V: RndA	Value: random challenge
Le	1,2	-	Length of expected response
		0x00/0x0000	Any expected length up to resp. 256/65536 bytes.
		0x56 .. 0xFFFF	Max expected length must be at least 86 bytes.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 79. Return code description - ISOInternalAuthenticate

Status	Value	Description
ISO9000	0x9000	Correct execution
ISO6700	0x6700	Wrong or inconsistent APDU length.
ISO6984	0x6984	ECC-based Card-Unilateral Authentication disabled via the key policy of the targeted key.
ISO6985	0x6985	Current state different from VCState.NotAuthenticated
		ECC-based Card-Unilateral Authentication disabled over NFC interface.
		ECC-based Card-Unilateral Authentication disabled over I^2C interface.
		KeyUsageCtrLimit enabled for targeted key has been reached.
ISO6987	0x6987	Expected DO missing.
ISO6A86	0x6A86	Wrong parameter P1: different from 0x00.
		Wrong parameter P2: RFU bits set.
ISO6A88	0x6A88	Wrong parameter P2: Key targeted by PrivKeyNo does not exist.
ISO6C00	0x6C00	Wrong Le: expected length insufficient for response data.

Table 80. Response data parameters description - ISOInternalAuthenticate - ISO9000

Name	Length	Value	Description
AuthDOHdr	2	-	Authentication Data Objects Header (TL)
		T: 0x7C	Tag
		L: 0x54	Length of subsequent Authentication Data Objects
RndB	18	-	Authentication Data Objects: random from PICC (TLV)
		T: 0x81	Tag
		L: 0x10	Length of Value field
		V: RndB	Value: random
Sig.B	66	-	Authentication Data Objects: signature from PICC (TLV)
		T: 0x82	Tag
		L: 0x40	Length of Value field
		V: Sig.B	Value: Sig.B = ECDSA_{Sign}(Priv.B, 0xF0F0 [|| OptsA] || RndB || RndA)

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.2.4 AuthenticateEV2First

The detailed description of this command can be found in Section 7.5.8.2.

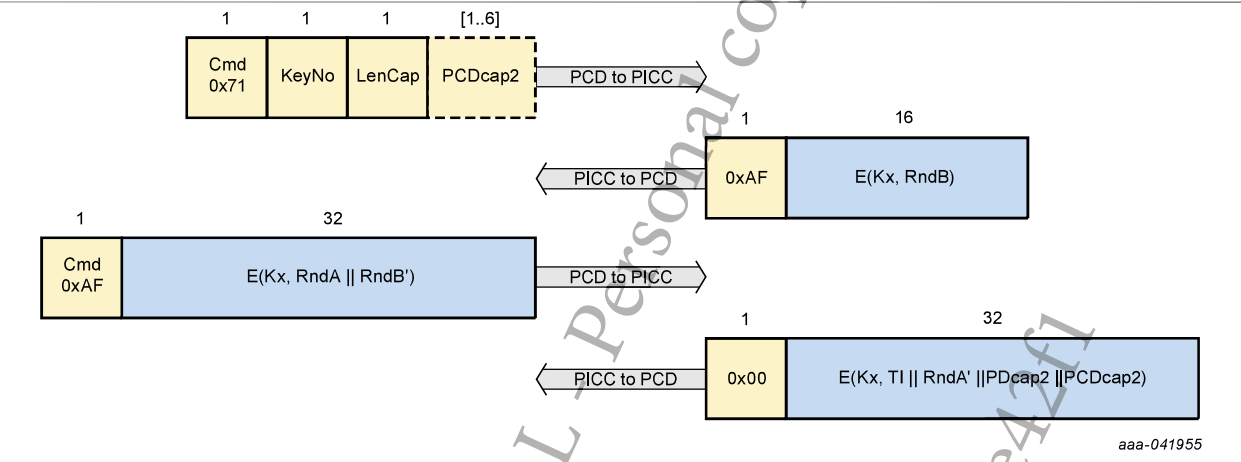


Figure 28. AuthenticateEV2First command protocol

Table 81. Command description - AuthenticateEV2First

AuthenticateEV2First	
Description:	Authentication for AES keys. After this authentication EV2 secure messaging is used. This authentication is intended to be the first in a transaction.
CommMode:	N/A (command specific)

Table 82. Command parameters description - AuthenticateEV2First

Name	Length	Value	Description
Command header parameters			
Cmd	1	0x71	Command code
KeyNo	1	-	Targeted authentication key
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6		RFU
		0b	
	Bit 5-0		Key Number
		Limited range	[if AID 0x000000 selected] At PICC level, key numbers are specified in Table 32 . This table also indicates which keys are available for authentication.
		0x0..0xD	[else] At application level, maximal key number depends on the number of application keys
LenCap	1	0x0..0x6	Length of the PCD Capabilities.
PCDcap2	[1..6]	Full range	Capability vector of the PCD.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07794-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 82. Command parameters description - AuthenticateEV2First...*continued*

Name	Length	Value	Description
Command data parameters			
-	-	-	No data parameters:

Table 83. Return code description - AuthenticateEV2First

Status	Value	Description
ADDITIONAL_FRAME	0xAF	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
NO_SUCH_KEY	0x40	Targeted key does not exist
PERMISSION_DENIED	0x9D	Targeted key not available for authentication.
		Targeted key not enabled.
		Targeting KeyID.NXPDAMAuthKey while PICCKeySettings Bit 4 is '0'.
		Targeting KeyID.PICCDAMAuthKey while PICCKeySettings Bit 4 is '1'.
		SAI given but no 2nd application selected.
		AES-based Symmetric Authentication disabled over NFC interface.
		AES-based Symmetric Authentication disabled over I^2C interface.
		AuthCtrLimit enabled for AES-based authentication has been reached.

Table 84. Response data parameters description - AuthenticateEV2First - ADDITIONAL_FRAME

Name	Length	Value	Description
E(Kx, RndB)	16	Full range	Encrypted PICC challenge

Table 85. Command description - AuthenticateEV2FirstPart2

AuthenticateEV2FirstPart2	
Description:	Second part of the first authentication.

Table 86. Command parameters description - AuthenticateEV2FirstPart2

Name	Length	Value	Description
Command header parameters			
Cmd	1	0xAF	Additional frame request.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 86. Command parameters description - AuthenticateEV2FirstPart2...*continued*

Name	Length	Value	Description
Command data parameters			
E(Kx, RndA || RndB')	32	-	Encrypted PCD challenge and response
		Full range	The following data, encrypted with the key Kx referenced by KeyNo: - RndA (16 byte): Random number of 16-byte from the PCD. - RndB' (16 byte): RndB rotated left over 1 byte.

Table 87. Return code description - AuthenticateEV2FirstPart2

Status	Value	Description
OPERATION_OK	0x00	
LENGTH_ERROR	0x7E	Command size not allowed.
AUTHENTICATION_ERROR	0xAE	Wrong RndB'

Table 88. Response data parameters description - AuthenticateEV2FirstPart2 - OPERATION_OK

Name	Length	Value	Description
E(Kx, TI || RndA' || PDcap2 || PCDcap2)	32	Full range	Encrypted PICC response

Provided under NDA only - Personal copy for:
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.2.5 AuthenticateEV2NonFirst

The detailed description of this command can be found in Section 7.5.8.3.

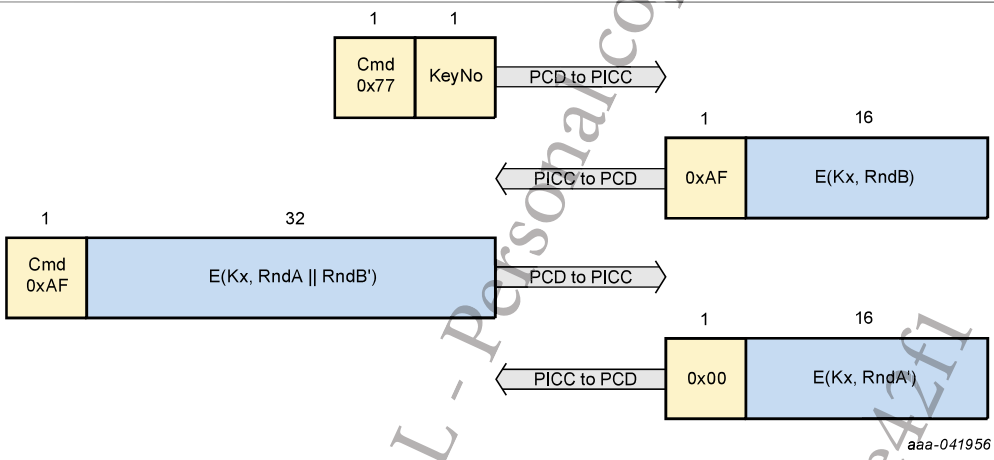


Figure 29. AuthenticateEV2NonFirst command protocol

Table 89. Command description - AuthenticateEV2NonFirst

AuthenticateEV2NonFirst	
Description:	Authentication for AES keys. After this authentication EV2 secure messaging is used. This authentication is intended for any subsequent authentication after AuthenticateEV2First in a transaction.
CommMode:	N/A (command specific)

Table 90. Command parameters description - AuthenticateEV2NonFirst

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x77	Command code.
KeyNo	1	-	Targeted authentication key
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6		RFU
		0b	
	Bit 5-0		Key Number
		Limited range	[if AID 0x000000 selected] At PICC level, key numbers are specified in Table 32 . This table also indicates which keys are available for authentication.
		0x0..0xD	[else] At application level, maximal key number depends on the number of application keys

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-22c-4055-90fe-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 90. Command parameters description - AuthenticateEV2NonFirst...*continued*

Name	Length	Value	Description
Command Data Parameters			
-	-	-	No data parameters:

Table 91. Return code description - AuthenticateEV2NonFirst

Status	Value	Description
ADDITIONAL_FRAME	0xAF	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing.
LENGTH_ERROR	0x7E	Command size not allowed.
PARAMETER_ERROR	0x9E	Parameter value not allowed.
NO_SUCH_KEY	0x40	Targeted key does not exist
PERMISSION_DENIED	0x9D	Not in VCState.AuthenticatedAES .
		Targeted key not available for authentication.
		Targeted key not enabled.
		Targeting KeyID.NXPDAMAuthKey while PICCKeySettings Bit 4 is '0'.
		Targeting KeyID.PICCDAMAuthKey while PICCKeySettings Bit 4 is '1'.
		SAI given but no 2nd application selected.
		AES-based Symmetric Authentication disabled over NFC interface.
		AES-based Symmetric Authentication disabled over I^2C interface.

Table 92. Response data parameters description - AuthenticateEV2NonFirst - ADDITIONAL_FRAME

Name	Length	Value	Description
E(Kx, RndB)	16	Full range	Encrypted PICC challenge

Table 93. Command description - AuthenticateEV2NonFirstPart2

AuthenticateEV2NonFirstPart2	
Description:	Second part of the non-first authentication.

Provided under NDA only
COMPANY CONFIDENTIAL - personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 94. Command parameters description - AuthenticateEV2NonFirstPart2

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xAF	Additional frame request.
Command Data Parameters			
E(Kx, RndA || RndB')	32	-	Encrypted PCD challenge and response
		Full range	The following data, encrypted with the key Kx referenced by KeyNo: - RndA (16 byte): Random number of 16-byte from the PCD. - RndB' (16 byte): RndB rotated left over 1 byte.

Table 95. Return code description - AuthenticateEV2NonFirstPart2

Status	Value	Description
OPERATION_OK	0x00	
LENGTH_ERROR	0x7E	Command size not allowed.
AUTHENTICATION_ERROR	0xAE	Wrong RndB'

Table 96. Response data parameters description - AuthenticateEV2NonFirstPart2 - OPERATION_OK

Name	Length	Value	Description
E(Kx, RndA')	16	Full range	Encrypted PICC response

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.3 Memory and configuration management

8.3.1 FreeMem

The detailed description of this command can be found in Section 7.7.4.1.

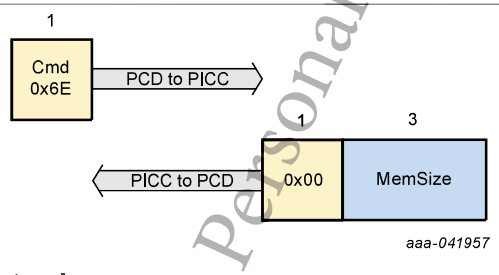


Figure 30. FreeMem command protocol

Table 97. Command description - FreeMem

FreeMem	
Description:	Returns the free memory available on the card
CommMode:	CommMode.MAC

Table 98. Command parameters description - FreeMem

Name	Length	Value	Description
Command header parameters			
Cmd	1	0x6E	Command code
Command data parameters			
-	-	-	No data parameters

Table 99. Return code description - FreeMem

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed

Table 100. Response data parameters description - FreeMem - OPERATION_OK

Name	Length	Value	Description
MemSize	3	Limited range	Size of the free memory

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07790-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.3.2 Format

The detailed description of this command can be found in Section 7.7.4.2.

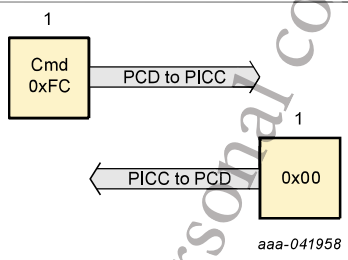


Figure 31. Format command protocol

Table 101. Command description - Format

Format	
Description:	At PICC level, all applications and files are deleted. At application level (only for delegated applications), all files are deleted. The deleted memory is released and can be reused.
CommMode:	CommMode.MAC

Table 102. Command parameters description - Format

Name	Length	Value	Description
Command header parameters			
Cmd	1	0xFC	Command code
Command data parameters			
-	-	-	No data parameters

Table 103. Return code description - Format

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
ILLEGAL_COMMAND_CODE	0x1C	At PICC level, format variant is disabled
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.PICCMasterKey at PICC level or KeyID.AppMasterKey (of primary application) at application level
CERT_ERROR	0xCE	Active ECC-based authentication not granting KeyID.PICCMasterKey access rights at PICC level or KeyID.AppMasterKey (of primary application) access rights at application level
PERMISSION_DENIED	0x9D	At application level, the targeted application is not a delegated application
		At application level, format is disabled for the targeted application

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
QuantaIT Solutions GmbH
Do not copy, share or reproduce
5fc07f9f-9ac-405c-90fe-6ecc3caef2f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.3.3 SetConfiguration

The detailed description of this command can be found in Section 7.7.3.1.

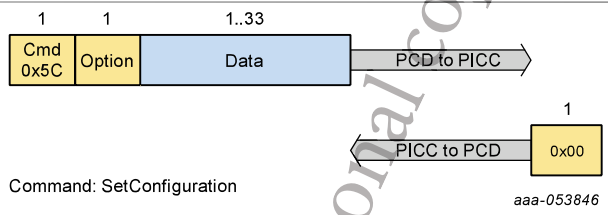


Figure 32. SetConfiguration command protocol

Table 104. Command description - SetConfiguration

SetConfiguration	
Description:	Configures several aspects of the card or the application
CommMode:	CommMode.Full

Table 105. Command parameters description - SetConfiguration

Name	Length	Value	Description
Command header parameters			
Cmd	1	0x5C	Command code
Option	1	-	Configuration option. Define length and content of the Data parameter
		0x00	PICC configuration
		0x01	Default keys update
		0x02	ATS Update
		0x03	SAK Update
		0x04	Secure Messaging Configuration
		0x05	Capability Data
		0x06	VC Installation Identifier
		0x07..0x0B	reserved
		0x0C	ATQA update
		0x0D	reserved
		0x0E	reserved
		0x0F	NFC management
		0x10	I^2C management
		0x11	GPIO Management
		0x12	reserved
		0x13	reserved
		0x14	reserved
		0x15	CryptoAPI management

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 105. Command parameters description - SetConfiguration...*continued*

Name	Length	Value	Description
		0x16	Authentication Counter and Limit Configuration
		0x17	HALT and Wake-up configuration
		0x18..0xFD	RFU
		0xFE	reserved
		0xFF	Lock configurations
Command Data Parameters:			
Data	1..33	-	Data content depends on option values
		Full range	Data content depends on option value as defined in Table 25

Table 106. Return code description - SetConfiguration

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid cryptogram (padding or CRC). Invalid secure messaging MAC (only EV2)
		Option 0x06: targeting delegated application with invalid DAMMAC
LENGTH_ERROR	0x7E	Command size not allowed
		Option 0x00: Data length is not 1 (only EV2 sec msg)
		Option 0x01: Data length is not 25 or 33
		Option 0x02: Data length is not in the range [1..20]
		Option 0x03: Data length is not 2
		Option 0x04: Data length is not 2 if SMConfig Bit 3 or Bit 6 is not set
		Option 0x04: Data length is not 3 if SMConfig Bit 3 or Bit 6 is set
		Option 0x05: Data length is not 10 (only EV2 sec msg)
		Option 0x06: Data inconsistent length (only EV2 sec msg)
		Option 0x0C: Data length is not 2
		Option 0x0F: Data length is not 3
		Option 0x10: Data length is not 4
		Option 0x11: Data length is not 33
		Option 0x15: Data length is not 2
		Option 0x16: Data length is not 6
		Option 0x17: Data length is not 4
		Option 0xFF: Data length is not 3.
PARAMETER_ERROR	0x9E	Parameter value not allowed

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-220c-465e-90fe-6ec c3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 106. Return code description - SetConfiguration...*continued*

Status	Value	Description
		Option 0x00: Data bit 7 not set to 0
		Option 0x00: Data bit 5 set to 1 and bit 1 set to 0
		Option 0x02: TL inconsistent with length of received ATS string
		Option 0x02: the FSCI value within T0 byte inside the ATS not supported by the PICC
		Option 0x05: invalid VCTID
		Option 0x06: invalid length or padding
		Option 0x11: unsupported GPIO1Mode, GPIO2Mode, GPIO1Notif, GPIO2Notif set
		Option 0x11: GPIO1Mode set to 0x03 not supported
		Option 0x11: AddAID set to 0x000000
		Option 0x15: unsupported Crypto API Support value different from 0x02
		Option 0x16: unsupported AuthCtrOption
		Option 0x17: unsupported configuration
PERMISSION_DENIED	0x9D	Targeted Option is locked.
		Option 0x00: issued at application level when targeting a conventional application
		Option 0x01: not supported at application level
		Option 0x02: not supported at application level
		Option 0x03: not supported at application level
		Option 0x04: enabling bit 4 (EV Charging support) is not supported at PICC level
		Option 0x04: enabling bit 5 (free access over I^2C) is not supported at PICC level
		Option 0x04: enabling bit 6 (fast confidential UID retrieval) is not supported at PICC level
		Option 0x04: enabling both bit 3 (ISOSelectFile integrated ECC-based authentication) and 6 (fast confidential UID retrieval) is not supported
		Option 0x04: enabling bit 3 (ISOSelectFile integrated ECC-based authentication) or bit 6 (fast confidential UID retrieval) is not supported for an application without ISOFileID and ISODFName
		Option 0x05: updating Capability data at application level while this has not been enabled at application creation
		Option 0x05: updating PDCap2 .2 to a value different than 0x00 at PICC level where the TransactionTimer is not supported
		Option 0x06: at application level, updating ISODFName while application was created without

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-0c-A05c-80fe-6ec5ae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 106. Return code description - SetConfiguration...*continued*

Status	Value	Description
		Option 0x0C: not supported at application level
		Option 0x0F: not supported at application level
		Option 0x0F: NFC support disabled by product configuration
		Option 0x10: not supported at application level
		Option 0x10: I^2C support disabled by product configuration
		Option 0x11: not supported at application level
		Option 0x11: GPIO support disabled by product configuration.
		Option 0x15: Not supported at application level if not enabled during application creation
		Option 0x17: Not supported at application level
AUTHENTICATION_ERROR	0xAE	Option 0x00, 0x01, 0x02, 0x03: No active authentication with KeyID.PICCMasterKey .
		Option 0x04: No active authentication with KeyID.PICCMasterKey at PICC level or KeyID.AppMasterKey at application level
		Option 0x05, 0x06: No active authentication with KeyID.VCConfigurationKey at PICC level or KeyID.AppMasterKey at application level
		Option 0x0C: No active authentication with KeyID.PICCMasterKey
		Option 0x0F: No active authentication with KeyID.PICCMasterKey .
		Option 0x10: No active authentication with KeyID.PICCMasterKey .
		Option 0x11: No active authentication with KeyID.PICCMasterKey .
		Option 0x15: No active authentication with KeyID.PICCMasterKey at PICC level or KeyID.AppMasterKey at application level
		Option 0x16: No active authentication KeyID.AppMasterKey at application level
		Option 0x16: No active authentication with KeyID.PICCMasterKey at PICC level
		Option 0x17: No active authentication with KeyID.PICCMasterKey at PICC level
		Option 0xFF: No active authentication with KeyID.PICCMasterKey
CERT_ERROR	0xCE	Option 0x00: Active ECC-based authentication not granting KeyID.PICCMasterKey access rights
		Option 0x01: Active ECC-based authentication not granting KeyID.PICCMasterKey access rights

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-0bc4-4bc9-9fe-66ec3e4ff

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 106. Return code description - SetConfiguration...*continued*

Status	Value	Description
		Option 0x02: Active ECC-based authentication not granting KeyID.PICCMasterKey access rights
		Option 0x03: Active ECC-based authentication not granting KeyID.PICCMasterKey access rights
		Option 0x04: Active ECC-based authentication not granting KeyID.PICCMasterKey access rights at PICC level or KeyID.AppMasterKey access rights at application level
		Option 0x05: Active ECC-based authentication not granting KeyID.VCConfigurationKey access rights at PICC level or KeyID.AppMasterKey access rights at application level
		Option 0x06: Active ECC-based authentication not granting KeyID.VCConfigurationKey access rights at PICC level or KeyID.AppMasterKey access rights at application level
		Option 0x0C: Active ECC-based authentication not granting KeyID.PICCMasterKey access rights
		Option 0x0F: Active ECC-based authentication not granting KeyID.PICCMasterKey access rights
		Option 0x10: Active ECC-based authentication not granting KeyID.PICCMasterKey access rights.
		Option 0x11: Active ECC-based authentication not granting KeyID.PICCMasterKey access rights
		Option 0x15: Active ECC-based authentication not granting KeyID.PICCMasterKey access rights at PICC level or KeyID.AppMasterKey access rights at application level
		Option 0x16: Active ECC-based authentication not granting KeyID.AppMasterKey at application level
		Option 0x16: Active ECC-based authentication not granting KeyID.PICCMasterKey at PICC level
		Option 0x17: Active ECC-based authentication not granting KeyID.PICCMasterKey at PICC level
		Option 0xFF: Active ECC-based authentication not granting KeyID.PICCMasterKey access rights
DUPLICATE_ERROR	0xDE	Option 0x06: renaming would result in duplicate DFNames.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-4b5-8ce-ecc2cde42a0

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.3.4 GetConfiguration

The detailed description of this command can be found in Section 7.7.3.2.

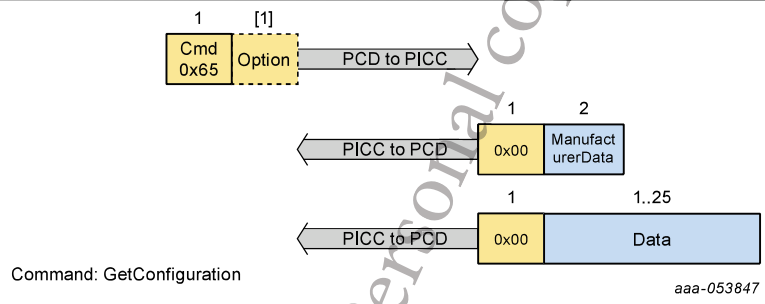


Figure 33. GetConfiguration command protocol

Table 107. Command description - GetConfiguration

GetConfiguration	
Description:	Retrieves configuration aspects of the card or the application.
CommMode:	CommMode.Full

Table 108. Command parameters description - GetConfiguration

Name	Length	Value	Description
Command header parameters			
Cmd	1	0x65	Command code
Option	[1]	-	Configuration option. If absent, manufacturer configuration data is returned
		Limited Range	For supported options, see SetConfiguration , except Option 0x01
Command data parameters			
-	-	-	No data parameters

Table 109. Return code description - GetConfiguration

Status	Value	Description
OPERATION_OK	0x00	[if no Option provided]
		[if Option provided]
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Unsupported Option
PERMISSION_DENIED	0x9D	Option not supported at the selected PICC or application level, see SetConfiguration
		Option 0x01 (KeyID.PICCAppDefaultKey) cannot be requested

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 109. Return code description - GetConfiguration...*continued*

Status	Value	Description
AUTHENTICATION_ERROR	0xAE	No active authentication with required key for the issued Option, see SetConfiguration
		No active authentication with KeyID.AppMasterKey or KeyID.PICCMasterKey if issued without Option
CERT_ERROR	0xCE	Active ECC-based authentication not granting access rights for the issued Option, see SetConfiguration
		Active ECC-based authentication not granting KeyID.AppMasterKey or KeyID.PICCMasterKey access rights if issued without Option

Table 110. Response data parameters description - GetConfiguration - OPERATION_OK

Name	Length	Value	Description
ManufacturerData	2	Full Range	Manufacturer Data

Table 111. Response data parameters description - GetConfiguration - OPERATION_OK

Name	Length	Value	Description
Data	1..25	-	Data content and length depends on option value as defined in Table 25

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.3.5 GetVersion

The detailed description of this command can be found in Section 7.7.2.1.

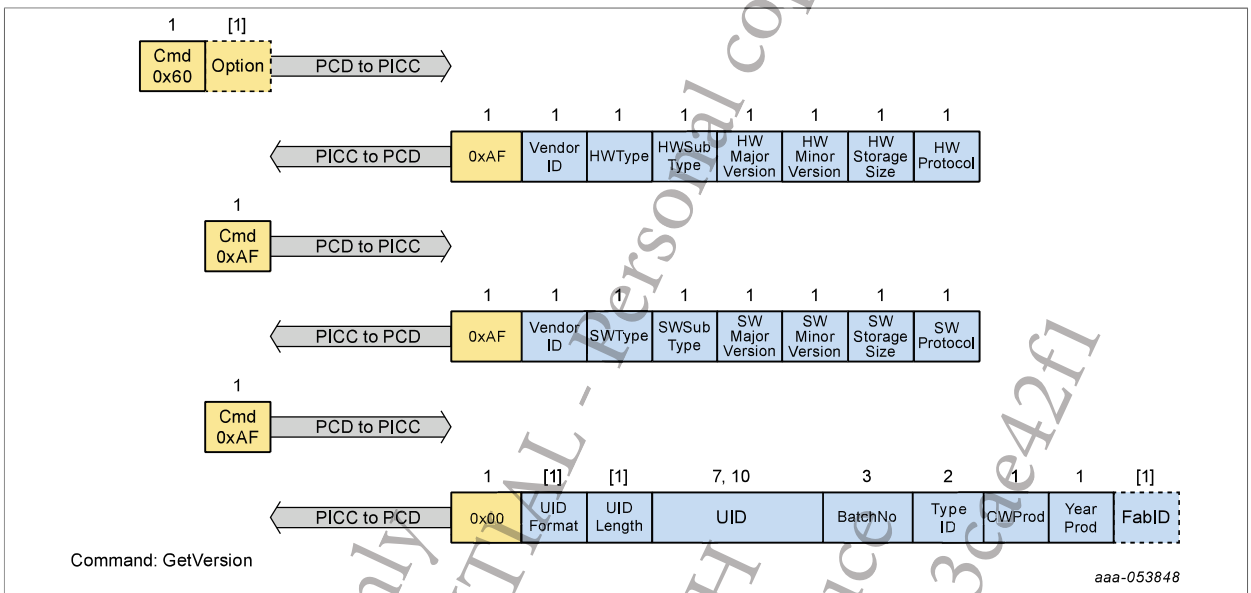


Figure 34. GetVersion command protocol

Table 112. Command description - GetVersion

GetVersion	
Description:	Returns manufacturing related data of the PICC. First part returns HW related information as specified in Table 24
CommMode:	CommMode.MAC

Table 113. Command parameters description - GetVersion

Name	Length	Value	Description
Command header parameters			
Cmd	1	0x60	Command code
Option	[1]	-	[Optional] Option byte
		0x01	Return Fab Identifier
Command data parameters			
-	-	-	No data parameters

Table 114. Return code description - GetVersion

Status	Value	Description
ADDITIONAL_FRAME	0xAF	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)

Provided under NDA only - Personal copy for: COMPANY CONFIDENTIAL Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-022a-05c-90fe-6ecc3ce42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 114. Return code description - GetVersion...*continued*

Status	Value	Description
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed

Table 115. Response data parameters description - GetVersion - ADDITIONAL_FRAME

Name	Length	Value	Description
VendorID	1	0x04	The vendor ID
HWType	1	0x01	The HW type
HWSubType	1	-	The HW subtype
		0x01	17 pF
		0x02	70 pF
HWMajorVersion	1	0xA0	The HW major version number
HWMinorVersion	1	0x00	The HW minor version number
HWStorageSize	1	-	The HW storage size
		0x16	2 kB
		0x18	4 kB
		0x1A	8 kB
		0x1C	16 kB
HWProtocol	1	-	The HW communication protocol type
		0x05	ISO/IEC 14443-4 support
		0x20	I^2C
		0x25	I^2C and ISO/IEC 14443-4 support

Table 116. Command description - GetVersionPart2

GetVersionPart2	
Description:	Returns manufacturing related data of the PICC. Second part returns SW related information as specified in Table 24

Table 117. Command parameters description - GetVersionPart2

Name	Length	Value	Description
Command header parameters			
Cmd	1	0xAF	Additional frame request
Command data parameters			
-	-	-	No data parameters

Provided under NDA only
COMPANY CONFIDENTIAL, Personal copy for:
Sergio Salate
Quantage IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-e2c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 118. Return code description - GetVersionPart2

Status	Value	Description
ADDITIONAL_FRAME	0xAF	
LENGTH_ERROR	0x7E	Command size not allowed

Table 119. Response data parameters description - GetVersionPart2 - ADDITIONAL_FRAME

Name	Length	Value	Description
VendorID	1	0x04	The vendor ID
SWType	1	0x01	The SW type
SWSubType	1	-	The SW subtype
	Bit 7-4		RFU
		0000b	
	Bit 3		
		0b	RFU
		1b	Released product configuration for product types listed in Section 4
	Bit 2		Legacy Secure Messaging
		1b	D40 and EV1 SM not supported
	Bit 1-0	11b	RFU
SWMajorVersion	1	0x00	The SW major version number
SWMinorVersion	1	-	The SW minor version number
	Bit 7-4		Minor version
		0x0	
	Bit 3-0		Sub Minor version
		0x1	Release version
SWStorageSize	1	-	The SW storage size
		0x16	2 kB
		0x18	4 kB
		0x1A	8 kB
		0x1C	16 kB
SWProtocol	1	-	The SW communication protocol type
		0x05	ISO/IEC 14443-4 support
		0x20	I^2C
		0x25	I^2C and ISO/IEC 14443-4 support

Table 120. Command description - GetVersionPart3

GetVersionPart3	
Description:	Returns manufacturing related data of the PICC. Third part returns production related information as specified in Table 24

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07779d-c22c-405c-90fe-6ecc3ae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 121. Command parameters description - GetVersionPart3

Name	Length	Value	Description
Command header parameters			
Cmd	1	0xAF	Additional frame request
Command data parameters			
-	-	-	No data parameters

Table 122. Return code description - GetVersionPart3

Status	Value	Description
OPERATION_OK	0x00	
LENGTH_ERROR	0x7E	Command size not allowed

Table 123. Response data parameters description - GetVersionPart3 - OPERATION_OK

Name	Length	Value	Description
UIDFormat	[1]	0x00	[Optional, present for non-7-byte UID] UID Format definition
UIDLength	[1]	0x0A	[Optional, present for non-7-byte UID] UID Length
UID	7, 10	-	[non-7-byte length only allowed if preceded by UIDFormat and correct UIDLength] VCUID
		All zero	if configured for RandomID (always 7 bytes returned)
		Full range	VCUID if not configured for RandomID
BatchNo	3	Full range	FabKey server batch number
TypeID	2	Limited range	Type identifier
CWProd	1	Full range	The calender week of production in BCD coding
YearProd	1	Full range	The year of production in BCD coding
FabID	[1]	Full range	[Optional, present if Option = 0x01] Fab Identifier

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.3.6 GetCardUID

The detailed description of this command can be found in Section 7.7.1.2.

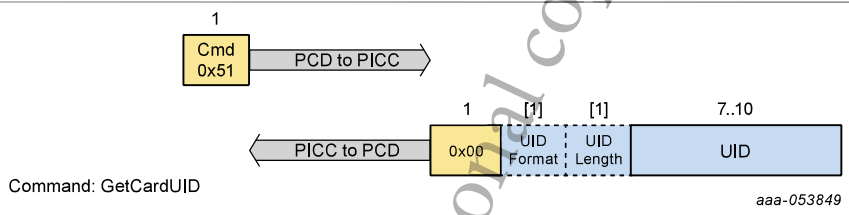


Figure 35. GetCardUID command protocol

Table 124. Command description - GetCardUID

GetCardUID	
Description:	Returns the UID
CommMode:	CommMode.Full

Table 125. Command parameters description - GetCardUID

Name	Length	Value	Description
Command header parameters			
Cmd	1	0x51	Command code
Command data parameters			
-	-	-	No data parameters

Table 126. Return code description - GetCardUID

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
AUTHENTICATION_ERROR	0xAE	No active authentication

Table 127. Response data parameters description - GetCardUID - OPERATION_OK

Name	Length	Value	Description
UIDFormat	[1]	0x00	UID Format definition
UIDLength	[1]	0x0A	[Present if UIDFormat == 0x00] UID Length
UID	7,10	Full Range	[non-7-byte length only allowed if preceded by UIDFormat and correct UIDLength] VCUID

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.4 Symmetric key management

8.4.1 ChangeKey

The detailed description of this command can be found in Section 7.8.6.1.

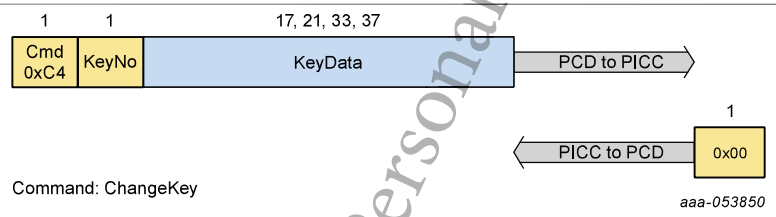


Figure 36. ChangeKey command protocol

Table 128. Command description - ChangeKey

ChangeKey	
Description:	Depending on the currently selected AID, this command update a key of the PICC or of an application AKS
CommMode:	N/A (command specific)

Table 129. Command parameters description - ChangeKey

Name	Length	Value	Description
Command header parameters			
Cmd	1	0xC4	Command code
KeyNo	1	-	Key number of the key to be changed. If current AID is 0x000000, also the new key type is given by this parameter
	Bit 7-6		[if AID 0x000000 is selected and targeting KeyID.PICCMasterKey] Key type
		00b	Reserved
		01b	Reserved
		10b	KeyType.AES128
		11b	KeyType.AES256
	Bit 7-6		[if AID 0x000000 is selected and targeting KeyID.VCConfigurationKey or KeyID.VCProximityKey] Key type
		'xx'	see above
	Bit 7-6		[if AID 0x000000 is not selected and targeting KeyID.VCProximityKey] Key type
		'xx'	see above
	Bit 7		[else] Second Application Indicator

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 129. Command parameters description - ChangeKey...*continued*

Name	Length	Value	Description
		0b	Target primary application
		1b	Target secondary application
	Bit 6		[else] RFU
		0b	
	Bit 5-0		Key Number
		Limited range	[if AID 0x000000 selected] At PICC level, key numbers are specified in Table 32
		0x0..0xD	[if AID 0x000000 is not selected] At application level, maximal key number depends on the number of application keys
		0x21	[if AID 0x000000 is not selected] Application specific KeyID.VCProximityKey if enabled at application creation
Command data parameters			
KeyData	17, 21, 33, 37	-	The new key
		Full range	[KeyNo == getKeyNo(AuthKey)] NewKey || KeyVer
		Full range	[KeyNo != getKeyNo(AuthKey)] (NewKey XOR OldKey) || KeyVer || CRC32NK

Table 130. Return code description - ChangeKey

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Integrity error in cryptogram or invalid secure messaging MAC (EV2 Secure Messaging)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
NO_SUCH_KEY	0x40	Targeted key does not exist
PERMISSION_DENIED	0x9D	At PICC level, targeting KeyID.NXPDAMAuthKey , KeyID.NXPDAMMACKey or KeyID.NXPDAMENCKey which cannot be changed
		At PICC level, targeting KeyID.PICCDAMAuthKey , KeyID.PICCDAMMACKey or KeyID.PICCDAMENCKey while the KeyID.NXPDAMKeys are enabled
		At PICC level, targeting KeyID.PICCMasterKey while PICCKeySettings [b 0] is set to 0
		At application level, targeted key type when changing KeyID.VCProximityKey set to KeyType.AES256 while not supported by the application

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c2-A055-90fe-6ec8cae2f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 130. Return code description - ChangeKey*...continued*

Status	Value	Description
		At application level, targeting KeyID.AppMasterKey while AppKeySettings [b 0] is set to 0
		At application level, targeting any key except KeyID.AppMasterKey while AppKeySettings [b 7..4] is set to 0xF
		SAI given but no 2nd application selected.
AUTHENTICATION_ERROR	0xAE	At PICC level, missing active authentication with KeyID.PICCMasterKey while targeting KeyID.PICCMasterKey , KeyID.PICCDAMAuthKey , KeyID.PICCDAMMACKey or KeyID.PICCDAMENCKey
		At PICC level, missing active authentication with KeyID.VCConfigurationKey while targeting KeyID.VCProximityKey
		At application level, missing active authentication with KeyID.AppMasterKey while targeting application specific KeyID.VCProximityKey
		At PICC level, missing active authentication with KeyID.PICCMasterKey or KeyID.VCConfigurationKey while targeting KeyID.VCConfigurationKey
		At application level, missing active authentication with KeyID.AppMasterKey while targeting KeyID.AppMasterKey , KeyID.AppRollKey or KeyID.AppChangeKey
		At application level if AppKeySettings [b 7..4] is set to [0x0..0xD], missing active authentication with this KeyID.AppChangeKey while targeting any key except KeyID.AppMasterKey , KeyID.AppRollKey or KeyID.AppChangeKey
		At application level if AppKeySettings [b 7..4] is set to 0xE, missing active authentication with the targeted key or its equivalent in the AKS while targeting any key except KeyID.AppMasterKey or KeyID.AppRollKey
CERT_ERROR	0xCE	Active ECC-based authentication not granting KeyID.PICCMasterKey access rights while targeting KeyID.PICCMasterKey , KeyID.PICCDAMAuthKey , KeyID.PICCDAMMACKey or KeyID.PICCDAMENCKey
		Active ECC-based authentication not granting KeyID.VCConfigurationKey access rights while targeting KeyID.VCProximityKey
		Active ECC-based authentication not granting KeyID.AppMasterKey access rights while targeting application specific KeyID.VCProximityKey
		Active ECC-based authentication not granting KeyID.PICCMasterKey or KeyID.VCConfigurationKey access rights while targeting KeyID.VCConfigurationKey
		Active ECC-based authentication not granting KeyID.AppMasterKey access rights while targeting KeyID.AppMasterKey , KeyID.AppRollKey or KeyID.AppChangeKey

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or modify
5fc0779d-2a2c-4b05-b2b0-6c5bae4211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 130. Return code description - ChangeKey...*continued*

Status	Value	Description
		Active ECC-based authentication not granting KeyID.AppChangeKey access rights, at application level if AppKeySettings [b 7..4] is set to [0x0..0xD], while targeting any key except KeyID.AppMasterKey, KeyID.AppRollKey or KeyID.AppChangeKey
		Active ECC-based authentication not granting access rights of the targeterd key, at application level if AppKeySettings [b 7..4] is set to 0xE, while targeting any key except KeyID.AppMasterKey or KeyID.AppRollKey

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.4.2 ChangeKeyEV2

The detailed description of this command can be found in Section 7.8.6.2.

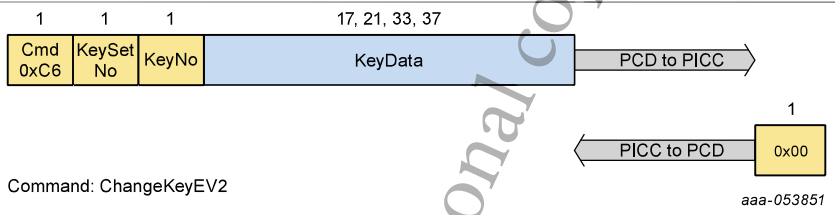


Figure 37. ChangeKeyEV2 command protocol

Table 131. Command description - ChangeKeyEV2

ChangeKeyEV2	
Description:	Depending on the currently selected AID, this command updates a key of the PICC or of one specified application keyset
CommMode:	N/A (command specific)

Table 132. Command parameters description - ChangeKeyEV2

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xC6	Command code
KeySetNo	1	-	Key set number
	Bit 7-4		RFU
		0x0	
	Bit 3-0		Key set number within targeted application
		0x0	[if AID 0x000000 is selected] No key sets at PICC level
		0x0	[if AID 0x000000 is not selected] At application level when not targeting regular applicaton keys (KeyNo outside 0x0..0x0D range)
		0x0..0xF	[else] Specified key set containing the targeted key
KeyNo	1	-	Key number of the key to be changed. If current AID is 0x000000, also the new key type is given by this parameter
	Bit 7-6		[if AID 0x000000 is selected and targeting KeyID.PICCMasterKey] Key type
		00b	Reserved
		01b	Reserved
		10b	KeyType.AES128
		11b	KeyType.AES256

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 132. Command parameters description - ChangeKeyEV2...*continued*

Name	Length	Value	Description
	Bit 7-6		[if AID 0x000000 is selected and targeting KeyID.VCConfigurationKey or KeyID.VCProximityKey] Key type
		'xx'	see above
	Bit 7-6		[if AID 0x000000 is not selected and targeting KeyID.VCProximityKey] Key type
		'xx'	see above
	Bit 7		[else] Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6		[else] RFU
		0b	
	Bit 5-0		Key Number
		Limited range	[if AID 0x000000 selected] At PICC level, key numbers are specified in Table 32
		0x0..0xD	[if AID 0x000000 is not selected] At application level, maximal key number depends on the number of application keys
		0x21	[if AID 0x000000 is not selected] Application specific KeyID.VCProximityKey if enabled at application creation
Command Data Parameters:			
KeyData	17, 21, 33, 37	-	The new key
		Full range	[KeyNo == getKeyNo(AuthKey)] NewKey || KeyVer
		Full range	[KeyNo != getKeyNo(AuthKey)] (NewKey XOR OldKey) || KeyVer || CRC32NK

Table 133. Return code description - ChangeKeyEV2

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Integrity error in cryptogram or invalid secure messaging MAC (EV2 Secure Messaging)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
NO_SUCH_KEY	0x40	Targeted key does not exist
		Targeted key set does not exist

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405f-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 133. Return code description - ChangeKeyEV2*...continued*

Status	Value	Description
PERMISSION_DENIED	0x9D	At PICC level, targeting KeyID.NXPDAMAuthKey, KeyID.NXPDAMMACKey or KeyID.NXPDAMENCKey which cannot be changed
		At PICC level, targeting KeyID.PICCDAMAuthKey, KeyID.PICCDAMMACKey or KeyID.PICCDAMENCKey while the KeyID.NXPDAMKeys are enabled
		At PICC level, targeting KeyID.PICCMasterKey while PICCKeySettings [b 0] is set to 0
		At application level, targeted key type when changing KeyID.VCProximityKey set to KeyType.AES256 while not supported by the application
		At application level, targeting KeyID.AppMasterKey while AppKeySettings [b 0] is set to 0
		At application level, targeting any key except KeyID.AppMasterKey while AppKeySettings [b 7..4] is set to 0xF
		SAI given but no 2nd application selected.
AUTHENTICATION_ERROR	0xAE	At PICC level, missing active authentication with KeyID.PICCMasterKey while targeting KeyID.PICCMasterKey , KeyID.PICCDAMAuthKey , KeyID.PICCDAMMACKey or KeyID.PICCDAMENCKey
		At PICC level, missing active authentication with KeyID.VCConfigurationKey while targeting KeyID.VCProximityKey .
		At application level, missing active authentication with KeyID.AppMasterKey while targeting application specific KeyID.VCProximityKey .
		At PICC level, missing active authentication with KeyID.PICCMasterKey or KeyID.VCConfigurationKey while targeting KeyID.VCConfigurationKey .
		At application level, missing active authentication with KeyID.AppMasterKey while targeting KeyID.AppMasterKey , KeyID.AppRollKey or KeyID.AppChangeKey

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions
Do not copy, share or upload
5fc0779d-c22c-4f2c-b0e3-cae6211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 133. Return code description - ChangeKeyEV2*...continued*

Status	Value	Description
		At application level if AppKeySettings [b 7..4] is set to [0x0..0xD], missing active authentication with this KeyID.AppChangeKey while targeting any key except KeyID.AppMasterKey , KeyID.AppRollKey or KeyID.AppChangeKey
		At application level if AppKeySettings [b 7..4] is set to 0xE, missing active authentication with the targeted key or its equivalent in the AKS while targeting any key except KeyID.AppMasterKey or KeyID.AppRollKey
CERT_ERROR	0xCE	Active ECC-based authentication not granting KeyID.PICCMasterKey access rights while targeting KeyID.PICCMasterKey , KeyID.PICCDAMAuthKey , KeyID.PICCDAMMACKey or KeyID.PICCDAMENCKey
		Active ECC-based authentication not granting KeyID.VCConfigurationKey access rights while targeting KeyID.VCProximityKey .
		Active ECC-based authentication not granting KeyID.AppMasterKey access rights while targeting application specific KeyID.VCProximityKey .
		Active ECC-based authentication not granting KeyID.PICCMasterKey or KeyID.VCConfigurationKey access rights while targeting KeyID.VCConfigurationKey
		Active ECC-based authentication not granting KeyID.AppMasterKey access rights while targeting KeyID.AppMasterKey , KeyID.AppRollKey or KeyID.AppChangeKey
		Active ECC-based authentication not granting KeyID.AppChangeKey access rights, at application level if AppKeySettings [b 7..4] is set to [0x0..0xD], while targeting any key except KeyID.AppMasterKey, KeyID.AppRollKey or KeyID.AppChangeKey
		Active ECC-based authentication not granting access rights of the targeterd key, at application level if AppKeySettings [b 7..4] is set to 0xE, while targeting any key except KeyID.AppMasterKey or KeyID.AppRollKey

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or distribute
5fc0779d-c22c-405c-ace-e3b6e4210

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.4.3 InitializeKeySet

The detailed description of this command can be found in Section 7.8.6.3.

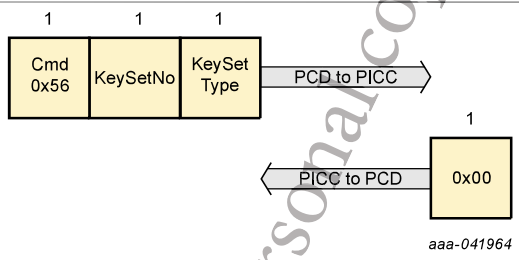


Figure 38. InitializeKeySet command protocol

Table 134. Command description - InitializeKeySet

InitializeKeySet	
Description:	Depending on the currently selected application, initialize the key set with specific index
CommMode:	CommMode.MAC

Table 135. Command parameters description - InitializeKeySet

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x56	Command code
KeySetNo	1	-	Key set number
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-4		RFU
		000b	
	Bit 3-0		Key set number within targeted application
		0x1..0xF	
KeySetType	1	-	Key type of the key set to be initialized
	Bit 7-2		RFU
		000000b	
	Bit 1-0		Key type of the application keys
		00b	Reserved
		01b	Reserved
		10b	KeyType.AES128
		11b	KeyType.AES256
Command Data Parameters			
-	-	-	No data parameters

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc079d-222c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 136. Return code description - InitializeKeySet

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	PICC level selected
		SAI given but no 2nd application selected
		Application keys not changeable as AppKeySettings [b 7..4] is set to 0xF
		Targeted key type is not supported by the application (due to disabled secure messaging mode or MaxKeySize during creation)
NO_SUCH_KEY	0x40	Targeted key set does not exist
AUTHENTICATION_ERROR	0xAE	If AppKeySettings [b 7..4] is set to [0x0..0xD], missing active authentication with this KeyID.AppChangeKey
		If AppKeySettings [b 7..4] is set to 0xE, missing active authentication with the KeyID.AppMasterKey
CERT_ERROR	0xCE	Active ECC-based authentication not granting KeyID.AppChangeKey access rights, if AppKeySettings [b 7..4] is set to [0x0..0xD]
		Active ECC-based authentication not granting KeyID.AppMasterKey access rights, if AppKeySettings [b 7..4] is set to 0xE

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.4.4 FinalizeKeySet

The detailed description of this command can be found in Section 7.8.6.4.

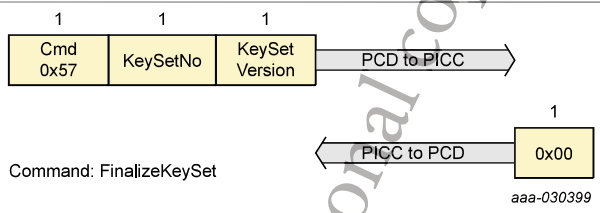


Figure 39. FinalizeKeySet command protocol

Table 137. Command description - FinalizeKeySet

FinalizeKeySet	
Description:	Within the currently selected application, finalize the key set with specified number
CommMode:	CommMode.MAC

Table 138. Command parameters description - FinalizeKeySet

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x57	Command code
KeySetNo	1	-	Key set number
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-4		RFU
		000b	
	Bit 3-0		Key set number within targeted application
		0x1..0xF	
KeySetVersion	1	0x01..0xFF	Key set version of the key set to be finalized
Command Data Parameters:			
-	-	-	No data parameters

Table 139. Return code description - FinalizeKeySet

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC.
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed

Provided under NDA only - Personal copy for: COMPANY CONFIDENTIAL Sergii Salata Quantum IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-405c-90f6-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 139. Return code description - FinalizeKeySet...*continued*

Status	Value	Description
PERMISSION_DENIED	0x9D	PICC level selected
		SAI given but no 2nd application selected
		Application keys not changeable as AppKeySettings [b 7..4] is set to 0xF
		Targeted key set has not key set version 0x00
NO_SUCH_KEY	0x40	Targeted key set does not exist
AUTHENTICATION_ERROR	0xAE	If AppKeySettings [b 7..4] is set to [0x0..0xD], missing active authentication with this KeyID.AppChangeKey
		If AppKeySettings [b 7..4] is set to 0xE, missing active authentication with the KeyID.AppMasterKey
CERT_ERROR	0xCE	Active ECC-based authentication not granting KeyID.AppChangeKey access rights, if AppKeySettings [b 7..4] is set to [0x0..0xD]
		Active ECC-based authentication not granting KeyID.AppMasterKey access rights, if AppKeySettings [b 7..4] is set to 0xE

Provided under NDA only - Personal copy for:
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3ca4a7f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.4.5 RollKeySet

The detailed description of this command can be found in Section 7.8.6.5.

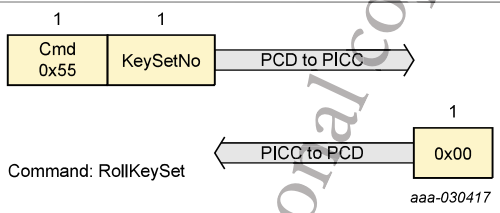


Figure 40. RollKeySet command protocol

Table 140. Command description - RollKeySet

RollKeySet	
Description:	Within the currently selected application, roll to the key set with specified number
CommMode:	CommMode.MAC

Table 141. Command parameters description - RollKeySet

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x55	Command code
KeySetNo	1	-	Key set number
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-4		RFU
		000b	
	Bit 3-0		Key set number within targeted application
		0x1..0xF	
Command Data Parameters			
-	-	-	No data parameters

Table 142. Return code description - RollKeySet

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	PICC level selected

Provided under NDA only - Personal copy for: COMPANY CONFIDENTIAL Sergii Solata Quantas IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-02c-105c-906c-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 142. Return code description - RollKeySet...*continued*

Status	Value	Description
		SAI given but no 2nd application selected
		Key set version of targeted key set is less than or equal to key set version of AKS
		Key type of targeted key set is not stronger than or equal to key type of AKS
NO_SUCH_KEY	0x40	Targeted key set does not exist
AUTHENTICATION_ERROR	0xAE	If AppKeySetSettings [b 3..0] is set to [0x0..0xD], missing active authentication with this KeyID.AppRollKey
CERT_ERROR	0xCE	Active ECC-based authentication not granting KeyID.AppRollKey access rights, if AppKeySetSettings [b 3..0] is set to [0x0..0xD]

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.4.6 GetKeySettings

The detailed description of this command can be found in Section 7.8.6.6.

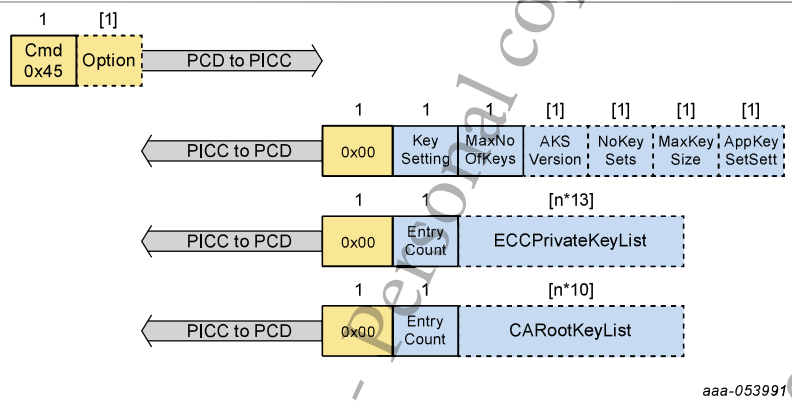


Figure 41. GetKeySettings command protocol

Table 143. Command description - GetKeySettings

GetKeySettings	
Description:	Depending on the currently selected AID, this command retrieves the PICCKeySettings of the PICC or the AppKeySettings of the (primary) application. In addition it returns the number of keys which are configured for the selected application and if applicable the AppKeySetSettings
Description:	This command retrieves the PICCKeySettings of the PICC, the AppKeySettings of the application, or meta-data of certain key types
CommMode:	CommMode.MAC

Table 144. Command parameters description - GetKeySettings

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x45	Command code
Option	[1]	-	
		0x01	KeyID.ECCPrivateKeys meta-data
		0x02	KeyID.CARootKeys meta-data
Command Data Parameters			
-	-	-	No data parameters

Table 145. Return code description - GetKeySettings

Status	Value	Description
OPERATION_OK	0x00	[No Option byte provided]
		[Option = 0x01] KeyID.ECCPrivateKeys meta-data
		[Option = 0x02] KeyID.CARootKeys meta-data
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing

Provided under NDA only
COMPANY CONFIDENTIAL - personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c2c405c-90fe-6ecccae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 145. Return code description - GetKeySettings*...continued*

Status	Value	Description
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
AUTHENTICATION_ERROR	0xAE	At PICC level, no active authentication with KeyID.PICCMasterKey or KeyID.PICCDAMAuthKey while PICCKeySettings [b 1] and PICCKeySettings [b 4] is not set
		At PICC level, no active authentication with KeyID.PICCMasterKey or KeyID.NXPDAMAuthKey while PICCKeySettings [b 1] is not set and PICCKeySettings [b 4] is set
		At application level, no active authentication with KeyID.AppMasterKey while AppKeySettings [b 1] is not set
CERT_ERROR	0xCE	Active ECC-based authentication not granting KeyID.PICCMasterKey or delegated application management access rights at PICC level while PICCKeySettings [b 1] and PICCKeySettings [b 4] is not set
		Active ECC-based authentication not granting KeyID.PICCMasterKey or delegated application management access rights at PICC level while PICCKeySettings [b 1] is not set and PICCKeySettings [b 4] is set
		Active ECC-based authentication not granting KeyID.AppMasterKey access rights at application level while AppKeySettings [b 1] is not set

Table 146. Response data parameters description - GetKeySettings - OPERATION_OK

Name	Length	Value	Description
KeySetting	1	-	
		Full range	[if AID 0x000000 selected] PICCKeySettings as in Table 33
		Full range	[else] AppKeySettings as in Table 36
MaxNoOfKeys	1	-	Maximum number of keys which can be stored within the selected application. Additionaly the key type is returned
	Bit 7-6		Key type as in Table 31
		00b	N/A, i.e. Bit 5-0 = 0x00
		01b	Reserved
		10b	KeyType.AES128
		11b	KeyType.AES256
	Bit 5-0		Number of keys
		0x1	[if AID 0x000000 is selected]
		0x0..0xD	[else] Number of application keys

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantum IT Solutions
Do not copy, share or reproduce
5fc07f0d-022-40fc-90fe-ecc6ca6a7

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 146. Response data parameters description - GetKeySettings - OPERATION_OK...*continued*

Name	Length	Value	Description
		0xF	[else] KeyID.AppMasterTmpKey enabled
AKSVersion	[1]	Full range	[if AID 0x000000 is not selected and the selected application has more than 1 key set] Key Set Version of the Active Key Set
NoKeySets	[1]	0x02..0x10	[if AID 0x000000 is not selected and the selected application has more than 1 key set] Number of Key Sets
MaxKeySize	[1]	-	[if AID 0x000000 is not selected and the selected application has more than 1 key set] Maximal key size
		0x10	KeyType.AES128 allowed (16 byte keys)
		0x20	KeyType.AES128 and KeyType.AES256 allowed (up to 32 byte keys)
AppKeySetSett	[1]	Full range	[if AID 0x000000 is not selected and the selected application has more than 1 key set] AppKeySetSettings as defined in Table 37

Table 147. Response data parameters description - GetKeySettings - OPERATION_OK - [Option = 0x01] KeyID.ECCPrivateKey s meta-data

Name	Length	Value	Description
EntryCount	1	0x00..0x05	Number of key information entries (n) that will follow
ECCPrivateKeyList	[n × 13]	-	List with meta-data, see ManageKeyPair
		Entry[0]	KeyNo
		Entry[1]	CurveID
		Entry[2..3]	KeyPolicy
		Entry[4]	WriteAccess
		Entry[5..8]	KeyUsageCtrLimit
		Entry[9..12]	KeyUsageCtr

Table 148. Response data parameters description - GetKeySettings - OPERATION_OK - [Option = 0x02] KeyID.CARootKeys meta-data

Name	Length	Value	Description
EntryCount	1	0x00..0x05	Number of key information entries (n) that will follow
CARootKeyList	[n × 10]	-	List with meta-data, see ManageCARootKey
		Entry[0]	KeyNo
		Entry[1]	CurveID
		Entry[2..3]	AccessRights

Provided under NDA only
COMPANY CONFIDENTIAL, Personal copy for:
Sergio Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5f0f9d1c-22c-405c-90fc-6ccc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 148. Response data parameters description - GetKeySettings - OPERATION_OK - [Option = 0x02] KeyID.CARootKeys meta-data...*continued*

Name	Length	Value	Description
		Entry[4]	WriteAccess
		Entry[5]	ReadAccess
		Entry[6]	CRLFile
		Entry[7..9]	CRLFileAID

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.4.7 ChangeKeySettings

The detailed description of this command can be found in Section 7.8.6.7.

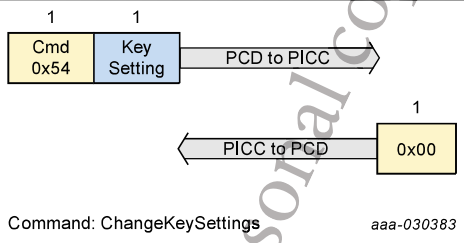


Figure 42. ChangeKeySettings command protocol

Table 149. Command description - ChangeKeySettings

ChangeKeySettings	
Description:	Depending on the currently selected AID, this command changes the PICCKeySettings of the PICC or the AppKeySettings of the application
CommMode:	CommMode.Full

Table 150. Command parameters description - ChangeKeySettings

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x54	Command code
Command Data Parameters			
KeySetting	1	-	
		Full range	[if AID 0x000000 selected] PICCKeySettings, see Table 33
		Full range	[else] AppKeySettings, see Table 36

Table 151. Return code description - ChangeKeySettings

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	Targeted PICCKeySettings or AppKeySettings not changeable
		At PICC level, PICCKeySettings [b 3] is not set
		At application level, AppKeySettings [b 3] is not set.
AUTHENTICATION_ERROR	0xAE	Missing authentication with KeyID.PICCMasterKey or KeyID.AppMasterKey

Provided under NDA only
COMPANY CONFIDENTIAL - personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 151. Return code description - ChangeKeySettings...*continued*

Status	Value	Description
		At PICC level, no active authentication with KeyID.PICCMasterKey
		At application level, no active authentication with KeyID.AppMasterKey
CERT_ERROR	0xCE	Active ECC-based authentication not granting KeyID.PICCMasterKey access rights at PICC level
		Active ECC-based authentication not granting KeyID.AppMasterKey access rights at application level
INTEGRITY_ERROR	0x1E	Integrity error in cryptogram. Invalid Secure Messaging MAC (only EV2)

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.4.8 GetKeyVersion

The detailed description of this command can be found in Section 7.8.6.8.

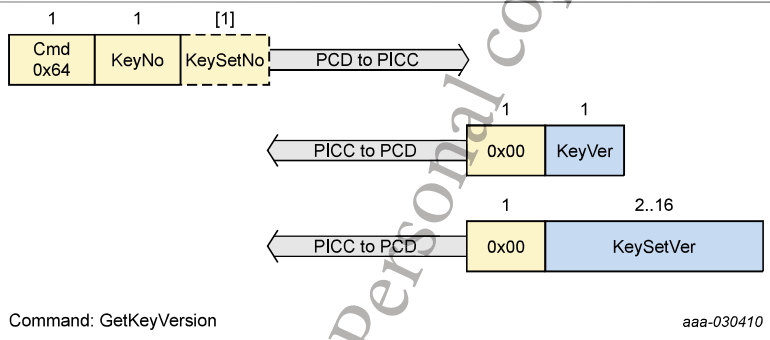


Figure 43. GetKeyVersion command protocol

Table 152. Command description - GetKeyVersion

GetKeyVersion	
Description:	Depending on the currently selected AID and given key number parameter, return key version of the key targeted or return all key set versions of the selected application
CommMode:	CommMode.MAC

Table 153. Command parameters description - GetKeyVersion

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x64	Command code
KeyNo	1	-	Key number of the targeted key
	Bit 7-6		[if AID 0x000000 is selected] RFU
		00b	
	Bit 7		[else] Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6		[else] KeySetNo presence
		0b	Not present
		1b	Present
	Bit 5-0		Key number of the targeted key
		Restricted Range	[if AID 0x000000 is selected] At PICC level, key numbers are specified in Table 32
		0x0	[if KeyNo[b6] and KeySetNo[b7] is set] RFU (Key set versions retrieval)

Provided under NDA only - Personal copy for: Sergi Salata
COMPANY CONFIDENTIAL
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22-405c-90fc-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 153. Command parameters description - GetKeyVersion...*continued*

Name	Length	Value	Description
		0x0..0xD	[else] At application level, maximal key number depends on the number of application keys
		0x21	[else] Application specific KeyID.VCProximityKey if available at application level)
KeySetNo	[1]	-	[Optional, present if KeyNo[b6] is set] Key set number
	Bit 7		Retrieval Target
		0b	Key version retrieval from specific key set
		1b	Key set versions retrieval
	Bit 6-4		RFU
		000b	
	Bit 3-0		Key set number within targeted application
		0x0	[if KeySetNo[b7] is set] RFU
		0x0..0xF	[if KeySetNo[b7] is not set] Specified key set containing the targeted key
Command Data Parameters			
-	-	-	No data parameters

Table 154. Return code description - GetKeyVersion

Status	Value	Description
OPERATION_OK	0x00	Key version retrieval
		Key set version retrieval (KeySetNo[b7] is set)
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	SAI given but no 2nd application selected
		At application level, KeyNo [b6] is set targeting key set related retrieval while the application has only one key set
		Targeted key is disabled
NO_SUCH_KEY	0x40	Targeted key does not exist (KeySetNo [b7] is not set)

Table 155. Response data parameters description - GetKeyVersion

Name	Length	Value	Description
KeyVer	1	Full range	Key version of the targeted key

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-22c4-0c90fe-6ec3ca42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 156. Response data parameters description - GetKeyVersion

Name	Length	Value	Description
KeySetVer	2..16	Full range	Key set versions of the selected application ordered by ascending key set number, i.e. starting with the AKS

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.5 Asymmetric Key Management

8.5.1 ManageKeyPair

The detailed description of this command can be found in Section 7.9.1.1.

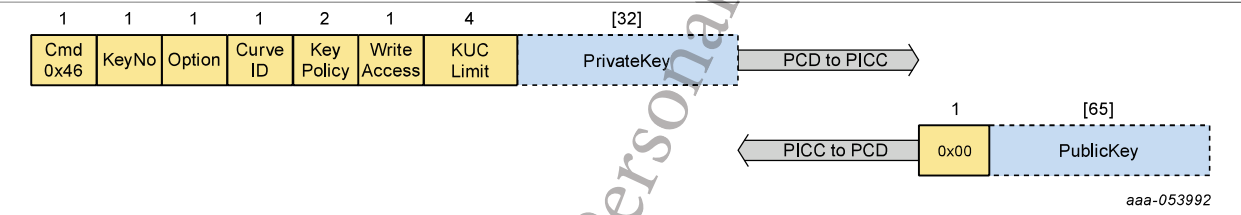


Figure 44. ManageKeyPair command protocol

Table 157. Command description - ManageKeyPair

ManageKeyPair	
Description:	Creates or updates a private key entry by generating a key pair or importing a private key
CommMode:	CommMode of targeted key, or if targeting not yet existing key CommMode.Full

Table 158. Command parameters description - ManageKeyPair

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x46	Command code
KeyNo	1	-	Key number of the key to be managed
	Bit 7-3		RFU
		00000b	
	Bit 2-0		KeyNo
		0x0..0x1	[if AID 0x000000 selected] At PICC level, two keys are supported
		0x0..0x4	[if AID 0x000000 is not selected] At application level, up to five keys are supported
Option	1	-	Targeted action
		0x00	Generate Key Pair
		0x01	Import Private Key
		0x02	Update Meta-Data
CurveID	1	-	Targeted curve
		0x0C	NIST P-256
		0x0D	brainpoolP256r1
KeyPolicy	2	-	Defines the allowed crypto operations with the targeted key.
	Bit 15		Freeze KeyUsageCtrLimit
		0b	disabled

Provided under NDA only - Personal copy for: COMPANY CONFIDENTIAL Sergi Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c2f2-49c-9016-6ecc3ca421

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 158. Command parameters description - ManageKeyPair...*continued*

Name	Length	Value	Description
		1b	enabled
	Bit 14-9		RFU
		000000b	
	Bit 8		ECC-based Card-Unilateral Authentication with Cmd.ISOInternalAuthenticate
		0b	disabled
		1b	enabled
	Bit 7		ECC-based Mutual Authentication
		0b	disabled
		1b	enabled
	Bit 6		ECC-based Transaction Signature
		0b	disabled
		1b	enabled
	Bit 5		ECC-based Secure Dynamic Messaging
		0b	disabled
		1b	enabled
	Bit 4		Cmd.CryptoRequest ECC Sign (Action 0x03)
		0b	disabled
		1b	enabled
	Bit 3-0		Reserved
		0000b	
WriteAccess	1	-	Defines the CommMode and access right required to update the key with Cmd.ManageKeyPair
	Bit 7-6		RFU
		00b	
	Bit 5-4		Write CommMode (see Table Table 7)
		'x0'	CommMode.Plain
		01b	CommMode.MAC
		11b	CommMode.Full
	Bit 3-0		WriteAR
		Full range	Access condition (see Table 16 or Table 17)
KUCLimit	4	-	Defines the key usage limit of the targeted key
		0x00000000	KeyUsageCtrLimit disabled
		0x00000001 .. 0xFFFFFFFF	KeyUsageCtrLimit enabled with the given value (LSB first)
Command Data Parameters			
PrivateKey	[32]	Full range	[Optional, present if Option is set to 0x01] Private key to be imported

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc079d-c22c-495c-90fe-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 159. Return code description - ManageKeyPair

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Integrity error in cryptogram or invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
NO_SUCH_KEY	0x40	Targeting non-existing key while trying to update meta-data
PERMISSION_DENIED	0x9D	Targeting existing key with its WriteAccess condition set to 0xF
		Trying to update meta-data, setting CurveID to a value different from the curve associated with the targeted key
		Trying to re-configure a frozen KeyUsageCtrLimit via update meta-data by setting Key Policy Bit 15 to '0' or KUCLimit to a different value than the currently configured KeyUsageCtrLimit
AUTHENTICATION_ERROR	0xAE	Targeting non-existing key while no active authentication with KeyID.PICCMasterKey at PICC level or KeyID.AppMasterKey at application level
		Targeting existing key while the WriteAccess condition of the targeted key different from 0xF not being granted.
CERT_ERROR	0xCE	Targeting non-existing key with an active ECC-based authentication not granting KeyID.PICCMasterKey access rights at PICC level or KeyID.AppMasterKey access rights at application level
		Targeting existing key with an active ECC-based authentication while the WriteAccess condition of the targeted key different from 0xF not being granted
OUT_OF_MEMORY_ERROR	0x0E	Insufficient free user memory available for creating new key

Table 160. Response data parameters description - ManageKeyPair - OPERATION_OK

Name	Length	Value	Description
PublicKey	[65]	Limited range	[Optional, present if Option is set to 0x00] Generated Public Key

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-9bfe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.5.2 ManageCARootKey

The detailed description of this command can be found in Section 7.9.2.1.

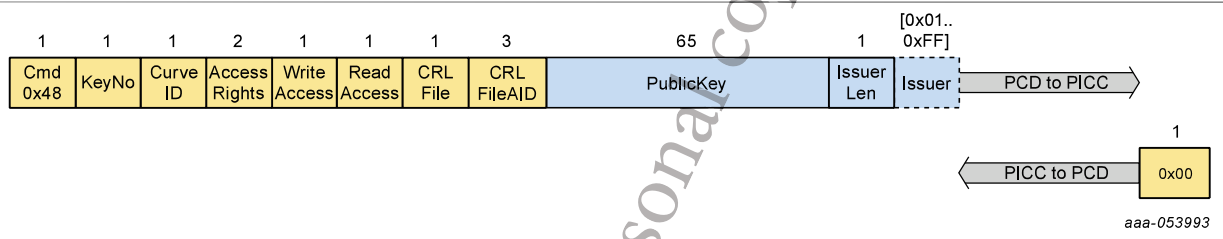


Figure 45. ManageCARootKey command protocol

Table 161. Command description - ManageCARootKey

ManageCARootKey	
Description:	Creates or updates a public key entry for storing a KeyID.CARootKey
CommMode:	CommMode of targeted key, or if targeting not yet existing key CommMode.Full

Table 162. Command parameters description - ManageCARootKey

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x48	Command code
KeyNo	1	-	Key number of the key to be managed
	Bit 7-3		RFU
		00000b	
	Bit 2-0		KeyNo
		0x0..0x1	[if AID 0x000000 selected] At PICC level, two keys are supported
		0x0..0x4	[if AID 0x000000 is not selected] At application level, up to five keys are supported
CurveID	1	-	Targeted curve
		0x0C	NIST P-256
		0x0D	brainpoolP256r1
AccessRights	2	Limited range	Access rights associated with the KeyID.CARootKey, see Table 18
WriteAccess	1	-	Defines the CommMode and access right required to update the key with Cmd.ManageCARootKey
	Bit 7-6		RFU
		00b	
	Bit 5-4		Write CommMode (see Table Table 7)
		'x0'	CommMode.Plain
		01b	CommMode.MAC

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-622c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 162. Command parameters description - ManageCARootKey...*continued*

Name	Length	Value	Description
		11b	CommMode.Full
	Bit 3-0		WriteAR
		Full range	Access condition (see Table 16 or Table 17)
ReadAccess	1	-	Defines the CommMode and access right required to read the key with Cmd.ExportKey
	Bit 7-6		RFU
		00b	
	Bit 5-4		Read CommMode (see Table Table 7)
		'x0'	CommMode.Plain
		01b	CommMode.MAC
		11b	CommMode.Full
	Bit 3-0		ReadAR
		Full range	Access condition (see Table 16 or Table 17)
CRLFile	1	-	Defines if certificate revocation is enabled and what file holds the CRL
	Bit 7		Enable certificate revocation
		0b	disabled
		1b	enabled
	Bit 6-5		RFU
		00b	
	Bit 4-0		CRL file
		0000b	[if Bit7 is '0'] RFU
		Limited range	[if Bit7 is '1'] File number
CRLFileAID	3	-	AID of the application holding the CRLFile
		0x000000	[if CRLFile Bit7 is '0'] RFU
		Full range	[if CRLFile Bit7 is '1'] AID
Command Data Parameters			
PublicKey	65	-	CA Public Key
		Limited range	Uncompressed public key: 0x04||Pub.x||Pub.y
IssuerLen	1	-	Length of trusted issuer name
		0x00	No trusted issuer name check required
		0x01..0xFF	Length of Issue.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22e-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Provided under NDA only - Personal copy for:
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-9b5e-aec5ae42f1

Table 162. Command parameters description - ManageCARootKey*...continued*

Name	Length	Value	Description
Issuer	[0x01.. 0xFF]	Full range	[Optional, present if IssuerLen != 0x00] Trusted issuer name of IssuerLen bytes

Table 163. Return code description - ManageCARootKey

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Integrity error in cryptogram or invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	Targeting existing key with its WriteAccess condition set to 0xF
AUTHENTICATION_ERROR	0xAE	Targeting non-existing key while no active authentication with KeyID.PICCMasterKey at PICC level or KeyID.AppMasterKey at application level
		Targeting existing key while the WriteAccess condition of the targeted key different from 0xF not being granted
CERT_ERROR	0xCE	Targeting non-existing key with an active ECC-based authentication not granting KeyID.PICCMasterKey access rights at PICC level or KeyID.AppMasterKey access rights at application level
		Targeting existing key with an active ECC-based authentication while the WriteAccess condition of the targeted key different from 0xF not being granted
OUT_OF_MEMORY_ERROR	0x0E	Insufficient free user memory available for creating new key

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.5.3 ExportKey

The detailed description of this command can be found in Section 7.9.3.

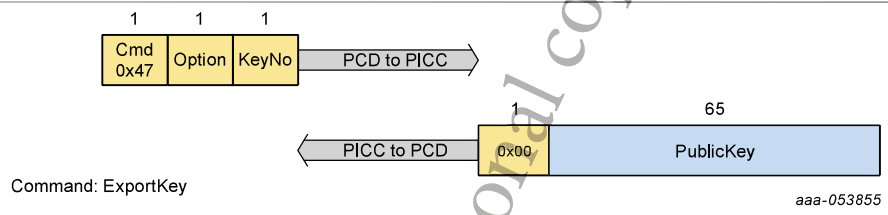


Figure 46. ExportKey command protocol

Table 164. Command description - ExportKey

ExportKey	
Description:	Exports the public key value of a KeyID.CARootKey
CommMode:	CommMode of targeted key

Table 165. Command parameters description - ExportKey

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x47	Command code
Option	1	-	Targeted key type
		0x01	Targeting KeyID.CARootKey
KeyNo	1	-	Key number of the key to be exported
	Bit 7-3		RFU
		00000b	
	Bit 2-0		KeyNo
		0x0..0x1	[if AID 0x000000 selected and and targeting KeyID.CARootKey] At PICC level, two keys are supported
		0x0..0x4	[if AID 0x000000 is not selected and targeting KeyID.CARootKey] At application level, up to five keys are supported
Command Data Parameters			
-	-	-	No data parameters

Table 166. Return code description - ExportKey

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Integrity error in cryptogram or invalid secure messaging MAC

Provided under NDA only - Personal copy for: COMPANY CONFIDENTIAL Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-06fc-90f6-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 166. Return code description - ExportKey*...continued*

Status	Value	Description
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
NO_SUCH_KEY	0x40	Targete key does not exist.
PERMISSION_DENIED	0x9D	Targeting existing key with its ReadAccess condition set to 0xF
AUTHENTICATION_ERROR	0xAE	Targeting existing key while the ReadAccess condition of the targeted key different from 0xF not being granted
CERT_ERROR	0xCE	Targeting existing key with an active ECC-based authentication while the ReadAccess condition of the targeted key different from 0xF not being granted

Table 167. Response data parameters description - ExportKey - OPERATION_OK

Name	Length	Value	Description
PublicKey	65	Limited range	Public Key

Provided under NDA only - Personal copy for:
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.5.4 GetKeySettings

The detailed description of this command can be found in Section 7.8.6.6.

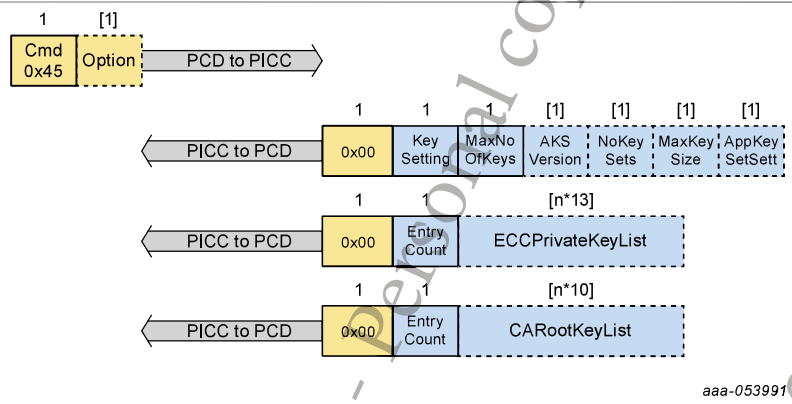


Figure 47. GetKeySettings command protocol

Table 168. Command description - GetKeySettings

GetKeySettings	
Description:	Depending on the currently selected AID, this command retrieves the PICCKeySettings of the PICC or the AppKeySettings of the (primary) application. In addition it returns the number of keys which are configured for the selected application and if applicable the AppKeySetSettings
Description:	This command retrieves the PICCKeySettings of the PICC, the AppKeySettings of the application, or meta-data of certain key types
CommMode:	CommMode.MAC

Table 169. Command parameters description - GetKeySettings

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x45	Command code
Option	[1]	-	
		0x01	KeyID.ECCPrivateKeys meta-data
		0x02	KeyID.CARootKeys meta-data
Command Data Parameters			
-	-	-	No data parameters

Table 170. Return code description - GetKeySettings

Status	Value	Description
OPERATION_OK	0x00	[No Option byte provided]
		[Option = 0x01] KeyID.ECCPrivateKeys meta-data
		[Option = 0x02] KeyID.CARootKeys meta-data
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c2c405c-90fe-6ecccae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 170. Return code description - GetKeySettings...*continued*

Status	Value	Description
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
AUTHENTICATION_ERROR	0xAE	At PICC level, no active authentication with KeyID.PICCMasterKey or KeyID.PICCDAMAuthKey while PICCKeySettings [b 1] and PICCKeySettings [b 4] is not set
		At PICC level, no active authentication with KeyID.PICCMasterKey or KeyID.NXPDAMAuthKey while PICCKeySettings [b 1] is not set and PICCKeySettings [b 4] is set
		At application level, no active authentication with KeyID.AppMasterKey while AppKeySettings [b 1] is not set
CERT_ERROR	0xCE	Active ECC-based authentication not granting KeyID.PICCMasterKey or delegated application management access rights at PICC level while PICCKeySettings [b 1] and PICCKeySettings [b 4] is not set
		Active ECC-based authentication not granting KeyID.PICCMasterKey or delegated application management access rights at PICC level while PICCKeySettings [b 1] is not set and PICCKeySettings [b 4] is set
		Active ECC-based authentication not granting KeyID.AppMasterKey access rights at application level while AppKeySettings [b 1] is not set

Table 171. Response data parameters description - GetKeySettings - OPERATION_OK

Name	Length	Value	Description
KeySetting	1	-	
		Full range	[if AID 0x000000 selected] PICCKeySettings as in Table 33
		Full range	[else] AppKeySettings as in Table 36
MaxNoOfKeys	1	-	Maximum number of keys which can be stored within the selected application. Additionaly the key type is returned
	Bit 7-6		Key type as in Table 31
		00b	N/A, i.e. Bit 5-0 = 0x00
		01b	Reserved
		10b	KeyType.AES128
		11b	KeyType.AES256
	Bit 5-0		Number of keys
		0x1	[if AID 0x000000 is selected]
		0x0..0xD	[else] Number of application keys

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantum IT Solutions
Do not copy, share or reproduce
5fc07f0d-022-40fc-90fe-ecc6ca6a21

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 171. Response data parameters description - GetKeySettings - OPERATION_OK...*continued*

Name	Length	Value	Description
		0xF	[else] KeyID.AppMasterTmpKey enabled
AKSVersion	[1]	Full range	[if AID 0x000000 is not selected and the selected application has more than 1 key set] Key Set Version of the Active Key Set
NoKeySets	[1]	0x02..0x10	[if AID 0x000000 is not selected and the selected application has more than 1 key set] Number of Key Sets
MaxKeySize	[1]	-	[if AID 0x000000 is not selected and the selected application has more than 1 key set] Maximal key size
		0x10	KeyType.AES128 allowed (16 byte keys)
		0x20	KeyType.AES128 and KeyType.AES256 allowed (up to 32 byte keys)
AppKeySetSett	[1]	Full range	[if AID 0x000000 is not selected and the selected application has more than 1 key set] AppKeySetSettings as defined in Table 37

Table 172. Response data parameters description - GetKeySettings - OPERATION_OK - [Option = 0x01] KeyID.ECCPrivateKey s meta-data

Name	Length	Value	Description
EntryCount	1	0x00..0x05	Number of key information entries (n) that will follow
ECCPrivateKeyList	[n × 13]	-	List with meta-data, see ManageKeyPair
		Entry[0]	KeyNo
		Entry[1]	CurveID
		Entry[2..3]	KeyPolicy
		Entry[4]	WriteAccess
		Entry[5..8]	KeyUsageCtrLimit
		Entry[9..12]	KeyUsageCtr

Table 173. Response data parameters description - GetKeySettings - OPERATION_OK - [Option = 0x02] KeyID.CARootKeys meta-data

Name	Length	Value	Description
EntryCount	1	0x00..0x05	Number of key information entries (n) that will follow
CARootKeyList	[n × 10]	-	List with meta-data, see ManageCARootKey
		Entry[0]	KeyNo
		Entry[1]	CurveID
		Entry[2..3]	AccessRights

Provided under NDA only
COMPANY CONFIDENTIAL, Personal copy for:
Sergio Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5f0f7d1c-22c-405c-90fe-6ccc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 173. Response data parameters description - GetKeySettings - OPERATION_OK - [Option = 0x02] KeyID.CARootKeys meta-data*...continued*

Name	Length	Value	Description
		Entry[4]	WriteAccess
		Entry[5]	ReadAccess
		Entry[6]	CRLFile
		Entry[7..9]	CRLFileAID

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.6 Application management

8.6.1 CreateApplication

The detailed description of this command can be found in Section 7.11.1.1.

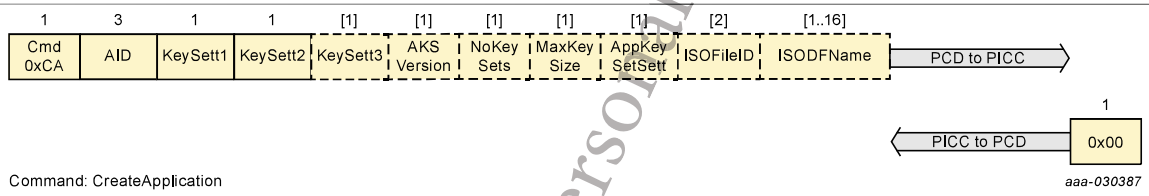


Figure 48. CreateApplication command protocol

Table 174. Command description - CreateApplication

CreateApplication	
Description:	Creates new applications on the PICC. The application is initialized according to the given settings. The application keys of the active key set are initialized with the Default Application Key.
CommMode:	CommMode.MAC

Table 175. Command parameters description - CreateApplication

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xCA	Command code
AID	3	-	Application Identifier of the application to be created
		0x000001 .. 0xFFFFFF	AID 0x000000 is reserved for the PICC level
KeySett1	1	Limited range	Application Key settings, see Table 36
KeySett2	1	-	Several other key settings
	Bit 7-6		KeyType of the application keys of the initial AKS
		00b	Reserved
		00b	Reserved
		10b	KeyType.AES128
		11b	KeyType.AES256
	Bit 5		Use of 2 byte ISO/IEC 7816-4 File Identifiers
		0b	No 2 byte File Identifiers for files within the application
		1b	2 byte File Identifiers for files within the application required
	Bit 4		KeySett3 presence
		0b	disabled
		1b	enabled
	Bit 3-0		Number of application keys (n)
		0x0..0xE	Maximum 14 keys

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 175. Command parameters description - CreateApplication...*continued*

Name	Length	Value	Description
		0xF	Enable KeyID.AppMasterTmpKey
KeySett3	[1]	-	[Optional, present if KeySett2[b4] is set] Additional optional key settings
	Bit 7-5		RFU
		000b	
	Bit 4		Application deletion with KeyID.AppMasterKey
		0b	depending on PICCKeySettings
		1b	always enabled
	Bit 3		Reserved
		0b	
	Bit 2		Application specific Capability data
		0b	disabled
		1b	enabled
	Bit 1		Application specific KeyID.VCProximityKey
		0b	disabled
		1b	enabled
	Bit 0		Application key sets
		0b	disabled
		1b	enabled
AKSVersion	[1]	Full range	[Optional, present if KeySett3[b0] is set] Key Set Version of the Active Key Set
NoKeySets	[1]	-	[Optional, present if KeySett3[b0] is set] Number of Key Sets
		0x02..0x10	Minimum 2 and maximum 16 key sets
MaxKeySize	[1]	-	[Optional, present if KeySett3[b0] is set] Max. Key Size Has to allow for the specified crypto method in KeySett2 Bit 7-6
		0x10	KeyType.AES128 allowed (16 byte keys)
		0x20	KeyType.AES128 and KeyType.AES256 allowed (up to 32 byte keys)
AppKeySetSett	[1]	Limited range	[Optional, present if KeySett3[b0] is set] Application Key Set Setting, see Table 37
ISOFileID	[2]	-	[Optional] ISO/IEC 7816-4 File Identifier for this application
		Full range	Excluding the following values reserved by ISO: 0x0000 0x3F00, 0x3FFF, 0xFFFF
ISODFName	[1..16]	Full range	[Optional, if present ISOFileID must also be present] ISO/IEC 7816-4 DF Name for this application
Command Data Parameters			
-	-	-	No data parameters

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Sala
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07790-c20c-4c3b-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 176. Return code description - CreateApplication

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
		Trying to enable key sets while enabling KeyID.AppMasterTmpKey
		Trying to enable key sets in an application with zero keys
		Trying to set AppKeySettings, Bit 7-4 different from 0x0 while enabling KeyID.AppMasterTmpKey
		Trying to set AppKeySettings, Bit 3 different from '0' while enabling KeyID.AppMasterTmpKey
		Trying to set AppKeySettings, Bit 0 different from '1' while enabling KeyID.AppMasterTmpKey
		Targeted MaxKeySize does not allow for AKS of KeyType.AES256 as targeted by KeySett2
PERMISSION_DENIED	0x9D	AID 0x000000 is not selected
		Targeted AKS key type is KeyType.AES256 while KeyID.PICCAppDefaultKey has been initialized with 24-byte value
AUTHENTICATION_ERROR	0xAE	No active KeyID.PICCMasterKey authentication while required by Bit 2 of the PICCKeySettings .
CERT_ERROR	0xCE	Active ECC-based authentication not granting KeyID.PICCMasterKey access rights while required by Bit 2 of the PICCKeySettings .
OUT_OF_MEMORY_ERROR	0x0E	Insufficient free user memory left for the application creation
DUPLICATE_ERROR	0xDE	An active application with the targeted AID already exists
		An active application with the targeted ISOFileID (if present) already exists
		An active application with the targeted ISODFName (if present) already exists
		The targeted ISODFName (if present) is already used at PICC level
		An active file with the targeted ISOFileID (if present) already exists at PICC level

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-40ac-9c0fe-6ec3cae251

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.6.2 DeleteApplication

The detailed description of this command can be found in Section 7.11.1.2.

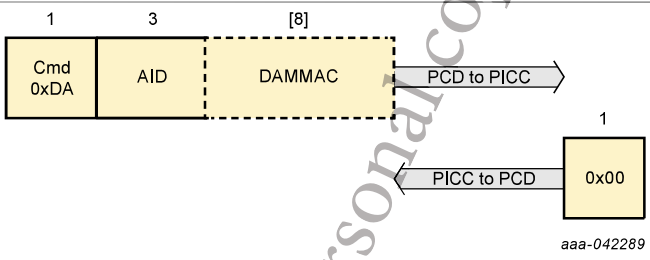


Figure 49. DeleteApplication command protocol

Table 177. Command description - DeleteApplication

DeleteApplication	
Description:	Permanently deactivates applications on the PICC. The AID is released
CommMode:	CommMode.MAC

Table 178. Command parameters description - DeleteApplication

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xDA	Command code
AID	3	-	Application Identifier of the application to be deleted
		0x000001 .. 0xFFFFFF	AID 0x000000 is reserved for the PICC level and cannot be deleted
DAMMAC	[8]	-	[Optional, present if KeyID.PICCDAMAuthKey or KeyID.NXPDAMAuthKey authentication] The MAC calculated by the card issuer to allow delegated application deletion
		Full range	DAMMAC as defined in Section 7.11.2.3.1
Command Data Parameters			
-	-	-	No data parameters

Table 179. Return code description - DeleteApplication

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
		Invalid DAMMAC
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 179. Return code description - DeleteApplication...*continued*

Status	Value	Description
PERMISSION_DENIED	0x9D	AID 0x000000 or targeted AID is not selected
		Targeting conventional application after KeyID.PICCDAMAuthKey or KeyID.NXPDAMAuthKey authentication
		Targeting conventional application after ECC-based authentication (eventually granting delegated application deletion access rights)
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.PICCMasterKey or KeyID.PICCDAMAuthKey while PICCKeySettings [b 2] is not set, KeySett3[b 4] was not set during creation, and PICCKeySettings [b 4] is not set
		No active authentication with KeyID.PICCMasterKey or KeyID.NXPDAMAuthKey while PICCKeySettings [b 2] is not set, KeySett3[b 4] was not set during creation, and PICCKeySettings [b 4] is set
		No active authentication with KeyID.PICCMasterKey or KeyID.AppMasterKey or KeyID.PICCDAMAuthKey while PICCKeySettings [b 4] is not set
		No active authentication with KeyID.PICCMasterKey or KeyID.AppMasterKey or KeyID.NXPDAMAuthKey while PICCKeySettings [b 4] is set.
CERT_ERROR	0xCE	Active ECC-based authentication, but not granting KeyID.PICCMasterKey or delegated application deletion access rights, while PICCKeySettings [b 2] is not set and KeySett3[b 4] was not set during creation
		Active ECC-based authentication, but not granting KeyID.PICCMasterKey or KeyID.AppMasterKey or delegated application deletion access rights
		Active ECC-based authentication granting delegated application deletion access rights, but not for the targeted AID
APPLICATION_NOT_FOUND	0xA0	Targeted AID does not exist

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or upload
5fc0779d-c22c-405c-9fe-c05ca421

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.6.3 CreateDelegatedApplication

The detailed description of this command can be found in Section 7.11.2.2.

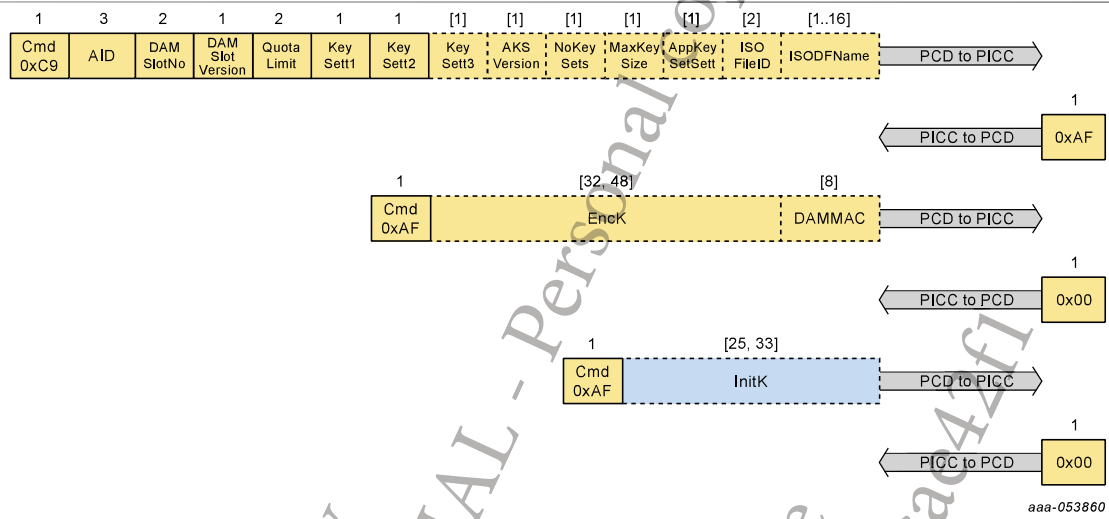


Figure 50. CreateDelegatedApplication command protocol

Table 180. Command description - CreateDelegatedApplication

CreateDelegatedApplication	
Description:	Creates delegated applications on the PICC with limited memory consumption. The application is initialized according to the given settings. The application keys of the active key set are initialized with the provided KeyID.AppDAMDefaultKey .
CommMode:	CommMode.Full

Table 181. Command parameters description - CreateDelegatedApplication

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xC9	Command code
AID	3	-	Application Identifier of the application to be created
		0x000001 .. 0xFFFFFF	AID 0x000000 is reserved for the PICC level
DAMSlotNo	2	Full range	Slot number associated with the memory space of the application to be created
DAMSlotVersion	1	Full range	Slot version associated with the memory space of the application to be created
QuotaLimit	2	Full range	Maximal memory consumption of the application in 32 byte blocks
KeySett1	1	Limited range	Application Key settings, see Table 36
KeySett2	1	Limited range	Several other key settings. For detailed definition, see CreateApplication

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cac42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 181. Command parameters description - CreateDelegatedApplication...*continued*

Name	Length	Value	Description
KeySett3	[1]	Limited range	[Optional, present if KeySett2[b4] is set] Additional optional key settings. For detailed definition, see CreateApplication
AKSVersion	[1]	Full range	[Optional, present if KeySett3[b0] is set] Key Set Version of the Active Key Set
NoKeySets	[1]	-	[Optional, present if KeySett3[b0] is set] Number of Key Sets
		0x02..0x10	Minimum 2 and maximum 16 key sets
MaxKeySize	[1]	-	[Optional, present if KeySett3[b0] is set] Max. Key Size Has to allow for the specified crypto method in KeySett2 Bit 7-6
		0x10	KeyType.AES128 allowed (16 byte keys)
		0x20	KeyType.AES128 and KeyType.AES256 allowed (up to 32 byte keys)
AppKeySetSett	[1]	Limited range	[Optional, present if KeySett3[b0] is set] Application Key Set Setting, see Table 37
ISOFileID	[2]	-	[Optional] ISO/IEC 7816-4 File Identifier for this application
		Full range	Excluding the following values reserved by ISO: 0x0000 0x3F00, 0x3FFF, 0xFFFF
ISODFName	[1..16]	Full range	[Optional, if present ISOFileID must also be present] ISO/IEC 7816-4 DF Name for this application
Command Data Parameters			
-	-	-	No data parameters

Table 182. Return code description - CreateDelegatedApplication

Status	Value	Description
ADDITIONAL_FRAME	0xAF	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
LENGTH_ERROR	0x7E	Command size not allowed

Table 183. Command description - CreateDelegatedApplicationPart2

CreateDelegatedApplicationPart2	
Description:	Second part of the CreateDelegatedApplication

Table 184. Command parameters description - CreateDelegatedApplicationPart2

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xAF	Additional frame request

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

Table 184. Command parameters description - CreateDelegatedApplicationPart2...*continued*

Name	Length	Value	Description
EncK	[32, 48]	-	[Optional, present if in VCState.AuthenticatedAES] The initial application key for the DAM application, encrypted with KeyID.PICCDAMENCKey
		Full range	EncK as defined in Section 7.11.2.2.1
DAMMAC	[8]	-	[Optional, present if in VCState.AuthenticatedAES] The MAC calculated by the card issuer with KeyID.PICCDAMMACKey to allow delegated application creation
		Full range	DAMMAC as defined in Section 7.11.2.2.2
Command Data Parameters			
InitK	[25, 33]	-	[Optional, present if in VCState.AuthenticatedECC] The initial application key for the DAM application: $K_{AppDAMDefaultKey}$|| KeyVer_{AppDAMDefault}
		Full range	In VCState.AuthenticatedECC, this key is handled as a data parameter and thus protected by the regular secure messaging CommMode.Full MACing and encryption

Table 185. Return code description - CreateDelegatedApplicationPart2

Status	Value	Description
OPERATION_OK	0x00	
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
		Invalid DAMMAC
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
		Trying to enable key sets while enabling KeyID.AppMasterTmpKey
		Trying to enable key sets in an application with zero keys
		Trying to set AppKeySettings. Bit 7-4 different from 0x0 while enabling KeyID.AppMasterTmpKey
		Trying to set AppKeySettings. Bit 3 different from '0' while enabling KeyID.AppMasterTmpKey
		Trying to set AppKeySettings. Bit 0 different from '1' while enabling KeyID.AppMasterTmpKey
		Targeted MaxKeySize does not allow for AKS of KeyType.AES256 as targeted by KeySett2

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-0a2c-4b8c-90fe-6ecc3ca6a7f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 185. Return code description - CreateDelegatedApplicationPart2...*continued*

Status	Value	Description
PERMISSION_DENIED	0x9D	AID 0x000000 is not selected
		Targeted DAMSlotNo already exists on card with equal or greater DAMSlotVersion and targeted DAMSlotVersion is different from 0xFF
		Targeted DAMSlotNo already exists on card with different QuotaLimit size
		KeyID.PICCDAMMACKey or KeyID.PICCDAMENCKey is disabled
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.PICCDAMAuthKey while PICCKeySettings [b 4] is not set
		No active authentication with KeyID.NXPDAMAuthKey while PICCKeySettings [b 4] is set
CERT_ERROR	0xCE	Active ECC-based authentication, but not granting delegated application creation access rights
		Active ECC-based authentication granting delegated application creation access rights, but not for the targeted application parameters
OUT_OF_MEMORY_ERROR	0x0E	QuotaLimit cannot be reserved
		QuotaLimit is insufficient for targeted application parameters
DUPLICATE_ERROR	0xDE	An active application with the targeted AID already exists and is not targeted to be replaced by this creation
		An active application with the targeted ISOFileID (if present) already exists and is not targeted to be replaced by this creation
		An active application with the targeted ISODFName (if present) already exists and is not targeted to be replaced by this creation
		The targeted ISODFName (if present) is already used at PICC level
		An active file with the targeted ISOFileID (if present) already exists at PICC level

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-45c-9fe-bec3ca2a21

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.6.4 SelectApplication

The detailed description of this command can be found in Section 7.11.3.2.

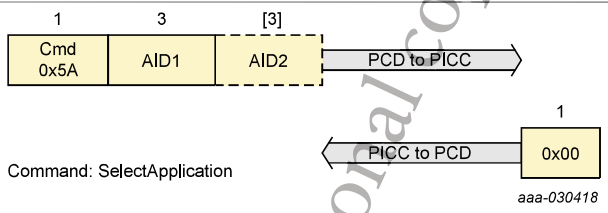


Figure 51. SelectApplication command protocol

Table 186. Command description - SelectApplication

SelectApplication	
Description:	Select 1 or 2 applications or the PICC level specified by their application identifier
CommMode:	N/A

Table 187. Command parameters description - SelectApplication

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x5A	Command code
AID1	3	Full range	Application Identifier of the primary application to be selected
AID2	[3]	Full range	[Optional] Application ID of the secondary application to be selected
Command Data Parameters			
-	-	-	No data parameters

Table 188. Return code description - SelectApplication

Status	Value	Description
OPERATION_OK	0x00	
OPERATION_OK_LIM	0x01	Application selected with limited functionality: TMCLimit reached
		Application selected with limited functionality: KeyID.AppTransactionSIGKey does not exist or is not enabled for Transaction Signature
		Application selected with limited functionality: KeyID.AppTransactionSIGKey has enabled KeyUsageCtrLimit reached
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
LENGTH_ERROR	0x7E	Command size not allowed

Provided under NDA only - Personal copy for: COMPANY CONFIDENTIAL Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-0a2c-405c-90fe-62cc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 188. Return code description - SelectApplication...*continued*

Status	Value	Description
PERMISSION_DENIED	0x9D	Two applications targeted and both applications are holding a FileType.TransactionMAC
		Two applications targeted and AID1 and AID2 are the same
		Targeted AID1 or AID2 is the PICC level (0x000000)
APPLICATION_NOT_FOUND	0xA0	Targeted AID1 or AID2 does not exist

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.6.5 GetApplicationIDs

The detailed description of this command can be found in Section 7.11.4.1.

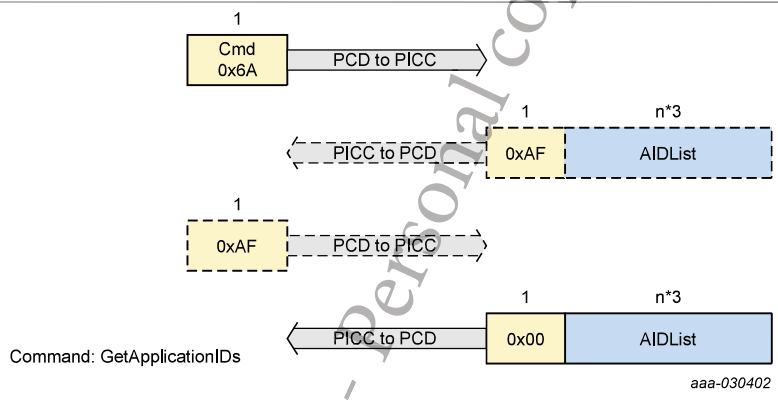


Figure 52. GetApplicationIDs command protocol

Table 189. Command description - GetApplicationIDs

GetApplicationIDs	
Description:	Returns the Application IDentifiers of all active applications on a PICC
CommMode:	CommMode.MAC

Table 190. Command parameters description - GetApplicationIDs

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x6A	Command code
Command Data Parameters			
-	-	-	No data parameters

Table 191. Return code description - GetApplicationIDs

Status	Value	Description
ADDITIONAL_FRAME	0xAF	Note that command chaining is only applied if the list does not fit into one response frame. In this case, the list is split between two AIDs, i.e. never a partial AID is sent
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PERMISSION_DENIED	0x9D	AID 0x000000 is not selected

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata, Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-2a2-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 191. Return code description - GetApplicationIDs*...continued*

Status	Value	Description
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.PICCMasterKey or KeyID.PICCDAMAuthKey while PICCKeySettings [b 1] and PICCKeySettings [b 4] is not set
		No active authentication with KeyID.PICCMasterKey or KeyID.NXPDAMAuthKey while PICCKeySettings [b 1] is not set and PICCKeySettings [b 4] is set
CERT_ERROR	0xCE	Active ECC-based authentication, but not granting KeyID.PICCMasterKey or delegated application management access rights

Table 192. Response data parameters description - GetApplicationIDs - ADDITIONAL_FRAME

Name	Length	Value	Description
AIDList	n × 3	Full Range	List of n 3-byte AIDs

Table 193. Response data parameters description - GetApplicationIDs - OPERATION_OK

Name	Length	Value	Description
AIDList	n × 3	Full Range	List of n 3-byte AIDs

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.6.6 GetDFNames

The detailed description of this command can be found in Section 7.11.4.2.

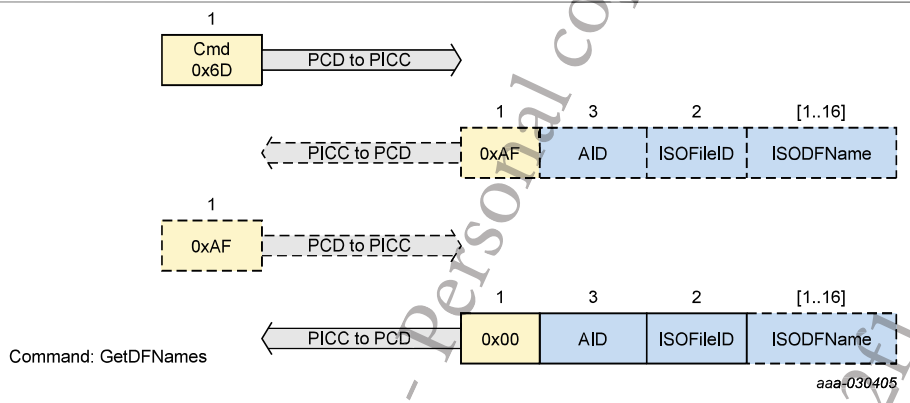


Figure 53. GetDFNames command protocol

Table 194. Command description - GetDFNames

GetDFNames	
Description:	Returns the the Application IDentifiers together with a File ID and (optionally) a DF Name of all active applications with ISO/IEC 7816-4 support
CommMode:	CommMode.MAC

Table 195. Command parameters description - GetDFNames

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x6D	Command code
Command Data Parameters			
-	-	-	No data parameters

Table 196. Return code description - GetDFNames

Status	Value	Description
ADDITIONAL_FRAME	0xAF	If there is more than one application with ISO/IEC 7816-4 support
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PERMISSION_DENIED	0x9D	AID 0x000000 is not selected

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salate
Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-22c4-405c-90fe-6ecc3cae4211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 196. Return code description - GetDFNames...*continued*

Status	Value	Description
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.PICCMasterKey or KeyID.PICCDAMAuthKey while PICCKeySettings [b 1] and PICCKeySettings [b 4] is not set
		No active authentication with KeyID.PICCMasterKey or KeyID.NXPDAMAuthKey while PICCKeySettings [b 1] is not set and PICCKeySettings [b 4] is set
CERT_ERROR	0xCE	Active ECC-based authentication, but not granting KeyID.PICCMasterKey or delegated application management access rights

Table 197. Response data parameters description - GetDFNames - ADDITIONAL_FRAME

Name	Length	Value	Description
AID	3	Full Range	Application Identifier
ISOFileID	2	Full Range	ISO/IEC 7816-4 File Identifier of this application
ISODFName	[1..16]	Full Range	ISO/IEC 7816-4 DF-Name of this application

Table 198. Response data parameters description - GetDFNames - OPERATION_OK

Name	Length	Value	Description
AID	3	Full Range	Application Identifier
ISOFileID	2	Full Range	ISO/IEC 7816-4 File Identifier of this application
ISODFName	[1..16]	Full Range	[Optional] ISO/IEC 7816-4 DF-Name of this application

Provided under NDA only for: COMPANY CONFIDENTIAL - personal copy for: Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-405c-9016-6ecc3ca42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.6.7 GetDelegatedInfo

The detailed description of this command can be found in Section 7.11.4.3.

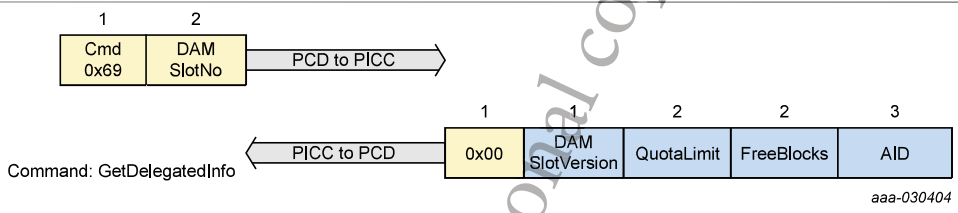


Figure 54. GetDelegatedInfo command protocol

Table 199. Command description - GetDelegatedInfo

GetDelegatedInfo	
Description:	Returns the DAMSlotVersion and QuotaLimit of a target DAM slot on the card
CommMode:	CommMode.MAC

Table 200. Command parameters description - GetDelegatedInfo

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x69	Command code.
DAMSlotNo	2	Full range	Slot number associated with the memory space of the targeted delegated application
Command Data Parameters			
-	-	-	No data parameters

Table 201. Return code description - GetDelegatedInfo

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PERMISSION_DENIED	0x9D	AID 0x000000 is not selected
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.PICCMasterKey or KeyID.PICCDAMAuthKey while PICCKeySettings [b 1] and PICCKeySettings [b 4] is not set
		No active authentication with KeyID.PICCMasterKey or KeyID.NXPDAMAuthKey while PICCKeySettings [b 1] is not set and PICCKeySettings [b 4] is set
CERT_ERROR	0xCE	Active ECC-based authentication, but not granting KeyID.PICCMasterKey or delegated application management access rights
APPLICATION_NOT_FOUND	0xA0	Targeted DAMSlotNo does not exist

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-02c-405c-901e-ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 202. Response data parameters description - GetDelegatedInfo - OPERATION_OK

Name	Length	Value	Description
DAMSlotVersion	1	Full Range	Slot version associated with the memory space of the targeted delegated application
QuotaLimit	2	Limited Range	Maximal memory consumption of the targeted delegated application (in 32-byte blocks)
FreeBlocks	2	Limited Range	Amount of unallocated memory of the targeted delegated application (in 32-byte blocks)
AID	3	Full Range	Application Identifier of the targeted DAM slot. This parameter is set to 0x000000 if the DAM slot targeted by DAMSlotNo exists on the card, but is currently not occupied with an application, i.e. the latest application has been deleted

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.7 File Management

8.7.1 CreateStdDataFile

The detailed description of this command can be found in Section 7.12.4.1.

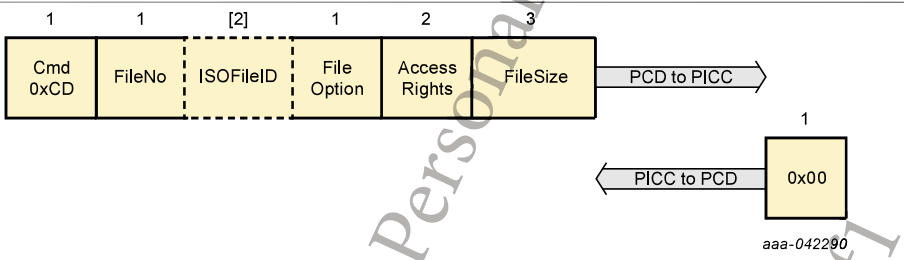


Figure 55. CreateStdDataFile command protocol

Table 203. Command description - CreateStdDataFile

CreateStdDataFile	
Description:	Creates files for the storage of plain unformatted user data within an existing application on the PICC.
CommMode:	CommMode.MAC

Table 204. Command parameters description - CreateStdDataFile

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xCD	Command code
FileNo	1	-	File number of the file to be created
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Limited range	
ISOFileID	[2]	-	[Optional] ISO/IEC 7816-4 File ID for the file to be created
		Full Range	Excluding the following values reserved by ISO: 0x0000 0x3F00, 0x3FFF, 0xFFFF
FileOption	1	-	Options for the targeted file
	Bit 7		Additional Access Rights
		0b	disabled
		1b	enabled
	Bit 6		Secure Dynamic Messaging and Mirroring

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405e-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 204. Command parameters description - CreateStdDataFile...*continued*

Name	Length	Value	Description
		0b	disabled
		1b	pre-enabled
	Bit 5		Reserved
		0b	
	Bit 4		CRL File
		0b	disabled
		1b	enabled
	Bit 3-2		RFU
		00b	
	Bit 1-0		CommMode (see Table Table 7)
		'x0'	CommMode.Plain
		01b	CommMode.MAC
		11b	CommMode.Full
AccessRights	2	Limited range	Set of access conditions for the 1st set in the file (see Table 41)
FileSize	3	-	File size in bytes for the file to be created
		0x00001 .. 0xFFFFFF	Empty file not allowed
Command Data Parameters			
-	-	-	No data parameters

Table 205. Return code description - CreateStdDataFile

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
		ISO/IEC 7816-4 File ID is enabled for the targeted application but not present in the received command
		ISO/IEC 7816-4 File ID is expected at PICC level but not present in the received command
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	SAI given but no 2nd application selected
		Trying to pre-enable SDM on File 0x1F
		Trying to pre-enable SDM at PICC level
		Trying to both pre-enable SDM and CRLFile
		Trying to enable additional access rights at PICC level

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 205. Return code description - CreateStdDataFile...*continued*

Status	Value	Description
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.AppMasterKey while required by AppKeySettings at application level
		No active authentication with KeyID.PICCMasterKey at PICC level
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting KeyID.AppMasterKey access rights while required from AppKeySettings at application level
		Active ECC-based authentication not granting KeyID.PICCMasterKey access rights at PICC leve
DUPLICATE_ERROR	0xDE	File with the targeted FileNo or ISOFileID already exists
		At PICC level, application with the targeted ISOFileID already exists
OUT_OF_MEMORY_ERROR	0x0E	Conventional application: insufficient free user memory available for creating this file
		Delegated application: QuotaLimit of targeted delegated application exceeded if creating this file

Provided under NDA only - Personal copy for:
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae421

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.7.2 CreateBackupDataFile

The detailed description of this command can be found in Section 7.12.4.2.

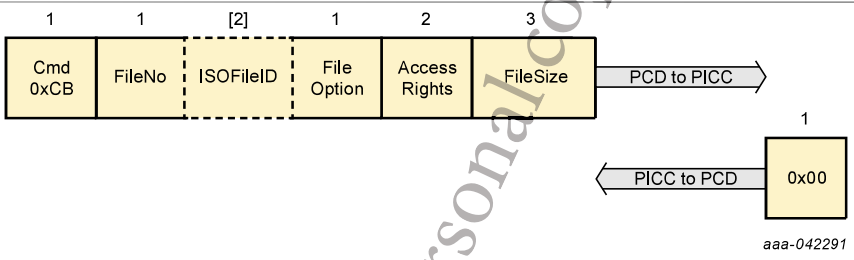


Figure 56. CreateBackupDataFile command protocol

Table 206. Command description - CreateBackupDataFile

CreateBackupDataFile	
Description:	Creates files for the storage of plain unformatted user data within an existing application on the PICC, additionally supporting the feature of an integrated backup mechanism
CommMode:	CommMode.MAC

Table 207. Command parameters description - CreateBackupDataFile

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xCB	Command code
FileNo	1	-	File number of the file to be created
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Limited range	
ISOFileID	[2]	-	[Optional] ISO/IEC 7816-4 File ID for the file to be created
		Full Range	Excluding the following values reserved by ISO: 0x0000 0x3F00, 0x3FFF, 0xFFFF
FileOption	1	-	Options for the targeted file.
	Bit 7		Additional Access Rights
		0b	disabled
		1b	enabled
	Bit 6		Reserved
		0b	
	Bit 5		Reserved

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 207. Command parameters description - CreateBackupDataFile...*continued*

Name	Length	Value	Description
		0b	
	Bit 4		CRL File
		0b	disabled
		1b	enabled
	Bit 3-2		RFU
		00b	
	Bit 1-0		CommMode (see Table Table 7)
		'x0'	CommMode.Plain
		01b	CommMode.MAC
		11b	CommMode.Full
AccessRights	2	Limited range	Set of access conditions for the 1st set in the file (see Table 41)
FileSize	3	-	File size in bytes for the file to be created
		0x00001 .. 0xFFFFFF	Empty file not allowed
Command Data Parameters:			
-	-	-	No data parameters

Table 208. Return code description - CreateBackupDataFile

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
		ISO/IEC 7816-4 File ID is enabled for the targeted application but not present in the received command
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		SAI given but no 2nd application selected
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.AppMasterKey while required by AppKeySettings
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting KeyID.AppMasterKey access rights while required from AppKeySettings
DUPLICATE_ERROR	0xDE	File with the targeted FileNo or ISOFileID already exists
OUT_OF_MEMORY_ERROR	0x0E	Conventional application: insufficient free user memory available for creating this file
		Delegated application: QuotaLimit of targeted delegated application exceeded if creating this file

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.7.3 CreateValueFile

The detailed description of this command can be found in Section 7.12.4.3.

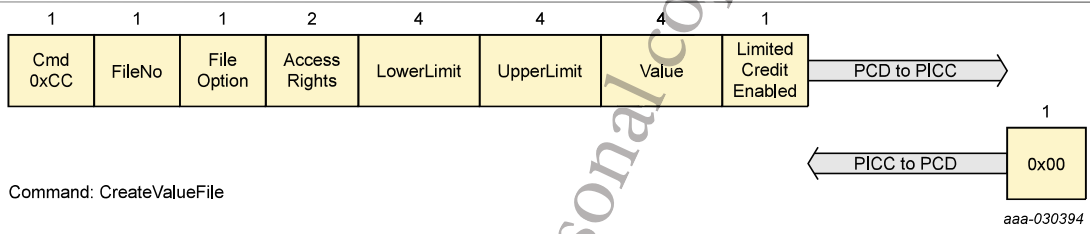


Figure 57. CreateValueFile command protocol

Table 209. Command description - CreateValueFile

CreateValueFile	
Description:	Creates files for the storage and manipulation of 32 bit signed integer values within an existing application on the PICC
CommMode:	CommMode.MAC

Table 210. Command parameters description - CreateValueFile

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xCC	Command code
FileNo	1	-	File number of the file to be created
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Limited range	
FileOption	1	-	Options for the targeted file
	Bit 7		Additional Access Rights
		0b	disabled
		1b	enabled
	Bit 6		RFU
		0b	
	Bit 5		Reserved
		0b	
	Bit 4-2		RFU
		000b	
	Bit 1-0		CommMode (see Table Table 7)

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-2a2c-405c-9ce6-ecc3cae42ff1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 210. Command parameters description - CreateValueFile...*continued*

Name	Length	Value	Description
		'x0'	CommMode.Plain
		01b	CommMode.MAC
		11b	CommMode.Full
AccessRights	2	Limited range	Set of access conditions for the 1st set in the file (see Table 41)
LowerLimit	4	-	Lower Limit of the file encoded as a 4 byte signed integer
		Full Range	with LowerLimit less than or equal to UpperLimit
UpperLimit	4	-	Upper Limit of the file encoded as a 4 byte signed integer
		Full Range	with LowerLimit less than or equal to UpperLimit
Value	4	-	Current Value.
		Full Range	with Value greater than or equal to LowerLimit and less than ore equal to UpperLimit
LimitedCreditEnabled	1	-	Encodes if LimitedCredit and free GetValue are allowed for this file.
	Bit 7-2		RFU
		000000b	
	Bit 1		Free GetValue
		0b	No free access to GetValue
		1b	Free access to GetValue
	Bit 0		LimitedCredit support
		0b	LimitedCredit disabled
		1b	LimitedCredit enabled
Command Data Parameters			
-	-	-	No data parameters

Table 211. Return code description - CreateValueFile

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
		Targeted key for one of the access conditions in AccessRights does not exist
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		SAI given but no 2nd application selected

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 211. Return code description - CreateValueFile...*continued*

Status	Value	Description
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.AppMasterKey while required by AppKeySettings
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting KeyID.AppMasterKey access rights while required from AppKeySettings
DUPLICATE_ERROR	0xDE	File with the targeted FileNo already exists
OUT_OF_MEMORY_ERROR	0x0E	Conventional application: insufficient free user memory available for creating this file
		Delegated application: QuotaLimit of targeted delegated application exceeded if creating this file

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.7.4 CreateLinearRecordFile

The detailed description of this command can be found in Section 7.12.4.4.

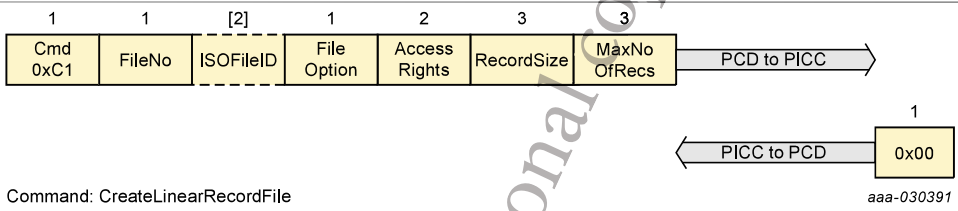


Figure 58. CreateLinearRecordFile command protocol

Table 212. Command description - CreateLinearRecordFile

CreateLinearRecordFile	
Description:	Creates files for multiple storage of structural similar data, for example for loyalty programs, within an existing application on the PICC. Once the file is filled completely with data records, further writing to the file is not possible unless it is cleared.
CommMode:	CommMode.MAC

Table 213. Command parameters description - CreateLinearRecordFile

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xC1	Command code
FileNo	1	-	File number of the file to be created
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Limited range	
ISOFileID	[2]	-	[Optional] ISO/IEC 7816-4 File ID for the file to be created
		Full Range	Excluding the following values reserved by ISO: 0x0000 0x3F00, 0x3FFF, 0xFFFF
FileOption	1	-	Options for the targeted file
	Bit 7		Additional Access Rights
		0b	disabled
		1b	enabled
	Bit 6-2		RFU
		00000b	
	Bit 1-0		CommMode (see Table Table 7)

Provided under NDA only
COMPANY CONFIDENTIAL. Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 213. Command parameters description - CreateLinearRecordFile...*continued*

Name	Length	Value	Description
		'x0'	CommMode.Plain
		01b	CommMode.MAC
		11b	CommMode.Full
AccessRights	2	Limited range	Set of access conditions for the 1st set in the file (see Table 41)
RecordSize	3	-	Size of a single record in the file
		0x00001 .. 0xFFFFFF	Empty record not allowed
MaxNoOfRecs	3	-	Maximum number of records within the file
		0x00001 .. 0xFFFFFF	Empty file not allowed
Command Data Parameters			
-	-	-	No data parameters

Table 214. Return code description - CreateLinearRecordFile

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
		ISO/IEC 7816-4 File ID is enabled for the targeted application but not present in the received command
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		SAI given but no 2nd application selected
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.AppMasterKey while required by AppKeySettings
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting KeyID.AppMasterKey access rights while required from AppKeySettings
DUPLICATE_ERROR	0xDE	File with the targeted FileNo or ISOFileID already exists
OUT_OF_MEMORY_ERROR	0x0E	Conventional application: insufficient free user memory available for creating this file
		Delegated application: QuotaLimit of targeted delegated application exceeded if creating this file

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-22ab-4f3c-9e0e-5ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.7.5 CreateCyclicRecordFile

The detailed description of this command can be found in Section 7.12.4.5.

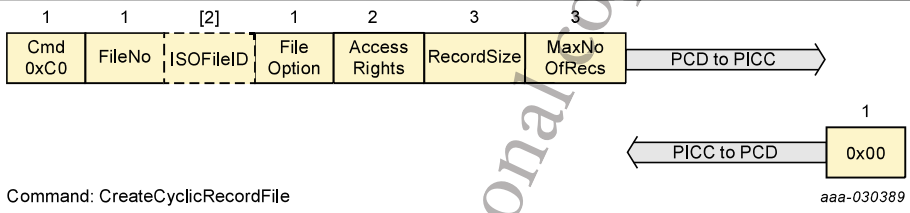


Figure 59. CreateCyclicRecordFile command protocol

Table 215. Command description - CreateCyclicRecordFile

CreateCyclicRecordFile	
Description:	Creates files for multiple storage of structural similar data, for example for logging transactions, within an existing application on the PICC. Once the file is filled completely with data records, the PICC automatically overwrites the oldest record with the latest written one. This wrap is fully transparent for the PCD.
CommMode:	CommMode.MAC

Table 216. Command parameters description - CreateCyclicRecordFile

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xC0	Command code
FileNo	1	-	File number of the file to be created
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Limited range	
ISOFileID	[2]	-	[Optional] ISO/IEC 7816-4 File ID for the file to be created
		Full Range	Excluding the following values reserved by ISO: 0x0000 0x3F00, 0x3FFF, 0xFFFF
FileOption	1	-	Options for the targeted file
	Bit 7		Additional Access Rights
		0b	disabled
		1b	enabled
	Bit 6-2		RFU
		00000b	
	Bit 1-0		CommMode (see Table Table 7)

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 216. Command parameters description - CreateCyclicRecordFile...*continued*

Name	Length	Value	Description
		'x0'	CommMode.Plain
		01b	CommMode.MAC
		11b	CommMode.Full
AccessRights	2	Limited range	Set of access conditions for the 1st set in the file (see Table 41)
RecordSize	3	-	Size of a single record in the file
		0x00001 .. 0xFFFFFF	Empty record not allowed
MaxNoOfRecs	3	-	Maximum number of records within the file
		0x00002 .. 0xFFFFFF	Empty file not allowed; one additional record for backup feature
Command Data Parameters			
-	-	-	No data parameters

Table 217. Return code description - CreateCyclicRecordFile

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
		ISO/IEC 7816-4 File ID is enabled for the targeted application but not present in the received command
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		SAI given but no 2nd application selected.
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.AppMasterKey while required by AppKeySettings
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting KeyID.AppMasterKey access rights while required from AppKeySettings
DUPLICATE_ERROR	0xDE	File with the targeted FileNo or ISOFileID already exists
OUT_OF_MEMORY_ERROR	0x0E	Conventional application: insufficient free user memory available for creating this file
		Delegated application: QuotaLimit of targeted delegated application exceeded if creating this file

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-22ff-4f3c-9ee-5ccc3cae4211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.7.6 CreateTransactionMACFile

The detailed description of this command can be found in Section 7.12.4.6.

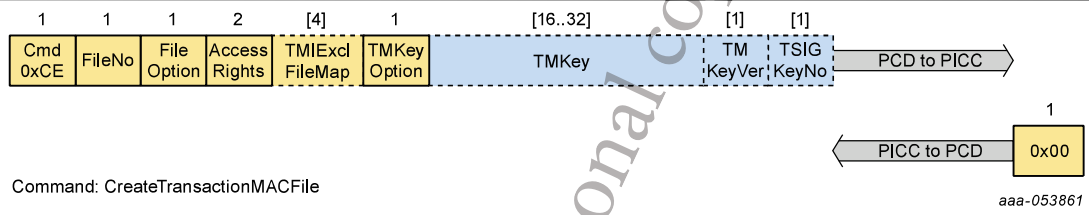


Figure 60. CreateTransactionMACFile command protocol

Table 218. Command description - CreateTransactionMACFile

CreateTransactionMACFile	
Description:	Creates a Transaction MAC File and enables the Transaction MAC or Signature feature for the targeted application.
CommMode:	CommMode.Full

Table 219. Command parameters description - CreateTransactionMACFile

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xCE	Command code
FileNo	1	-	File number of the file to be created
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Full Range	
FileOption	1	-	Options for the targeted file
	Bit 7		RFU
		0b	
	Bit 6		TMI exclusion filemap
		0b	disabled
		1b	enabled
	Bit 5-2		RFU
		0000b	
	Bit 1-0		CommMode (see Table Table 7)
		'x0'	CommMode.Plain
		01b	CommMode.MAC

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 219. Command parameters description - CreateTransactionMACFile...*continued*

Name	Length	Value	Description
		11b	CommMode.Full
AccessRights	2	-	Set of access conditions (see Table Table 40)
TMIExclFileMap	[4]	Full range	[Optional, present if FileOption[b6] set] TMI exclusion filemap
TMKeyOption	1	-	Options for KeyID.AppTransactionMACKey
	Bit 7		Mode
		0b	Transaction MAC
		1b	Transaction Signature
	Bit 6-2		RFU
		00000b	
	Bit 1-0		KeyType of the KeyID.AppTransactionMACKey
		00b	No KeyID.AppTransactionMACKey
		01b	RFU
		10b	KeyType.AES128
		11b	KeyType.AES256
Command Data Parameters			
TMKey	[16, 32]	Full Range	[Optional, present if Transaction MAC or authenticated CommitReaderID] Key value of KeyID.AppTransactionMACKey
TMKeyVer	[1]	Full Range	[Optional, present if Transaction MAC or authenticated CommitReaderID] Key version of KeyID.AppTransactionMACKey
TSIGKeyNo	[1]	-	[Optional, present if Transaction Signature] KeyID.AppTransactionSIGKey
	Bit 7-3		RFU
		00000b	
	Bit 2-0		KeyNo of targeted KeyID.ECCPrivateKey
		0x0..0x04	

Table 220. Return code description - CreateTransactionMACFile

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
		Trying to enable CommitReaderID without providing a KeyID.AppTransactionMACKey

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantagit Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c2c-3c-405c-90fe-ecc3cae40f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 220. Return code description - CreateTransactionMACFile...*continued*

Status	Value	Description
NO_SUCH_KEY	0x40	Targeted KeyID.ECCPrivateKey does not exist
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		One of the selected applications already holds a FileType.TransactionMAC file
		SAI given but no 2nd application selected
		Targeted KeyID.ECCPrivateKey does not support Transaction Signature
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.AppMasterKey while required by AppKeySettings
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting KeyID.AppMasterKey access rights while required from AppKeySettings
DUPLICATE_ERROR	0xDE	File with the targeted FileNo already exists
OUT_OF_MEMORY_ERROR	0x0E	Conventional application: insufficient free user memory available for creating this file
		Delegated application: QuotaLimit of targeted delegated application exceeded if creating this file

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3ca42f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.7.7 DeleteFile

The detailed description of this command can be found in Section 7.12.5.

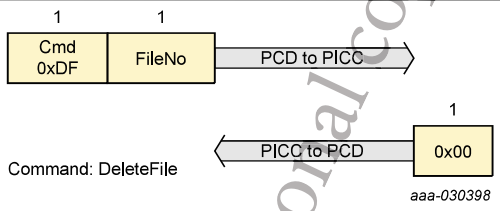


Figure 61. DeleteFile command protocol

Table 221. Command description - DeleteFile

DeleteFile	
Description:	Permanently deactivates a file within the file directory of the currently selected application
CommMode:	CommMode.MAC

Table 222. Command parameters description - DeleteFile

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xDF	Command code
FileNo	1	-	File number of the file to be deleted
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Full Range	
Command Data Parameters			
-	-	-	No data parameters

Table 223. Return code description - DeleteFile

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	SAI given but no 2nd application selected

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 223. Return code description - DeleteFile*...continued*

Status	Value	Description
FILE_NOT_FOUND	0xF0	Targeted file does not exist in the targeted application
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.AppMasterKey while required by AppKeySettings at application level
		No active authentication with KeyID.PICCMasterKey at PICC level
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting KeyID.AppMasterKey access rights while required from AppKeySettings at application level
		Active ECC-based authentication, but CertAccessRights are not granting KeyID.PICCMasterKey access rights at PICC level

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.7.8 GetFileIDs

The detailed description of this command can be found in Section 7.12.3.3.

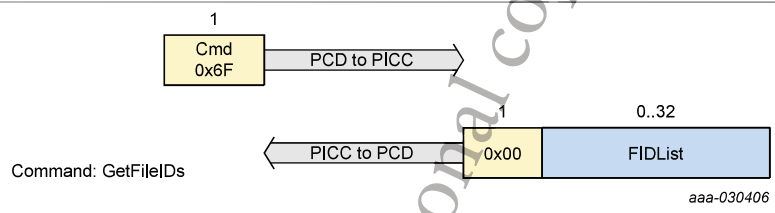


Figure 62. GetFileIDs command protocol

Table 224. Command description - GetFileIDs

GetFileIDs	
Description:	Returns the File IDentifiers of all active files within the currently selected application
CommMode:	CommMode.MAC

Table 225. Command parameters description - GetFileIDs

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x6F	Command code
Command Data Parameters			
-	-	-	No data parameters

Table 226. Return code description - GetFileIDs

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.AppMasterKey while required from AppKeySettings
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting KeyID.AppMasterKey access rights while required from AppKeySettings

Table 227. Response data parameters description - GetFileIDs - OPERATION_OK

Name	Length	Value	Description
FIDList	0..32	0x00..0x1F	List of n File IDs

Provided under NDA only - Personal copy for: COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07f9d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.7.9 GetISOFileIDs

The detailed description of this command can be found in Section 7.12.3.4.

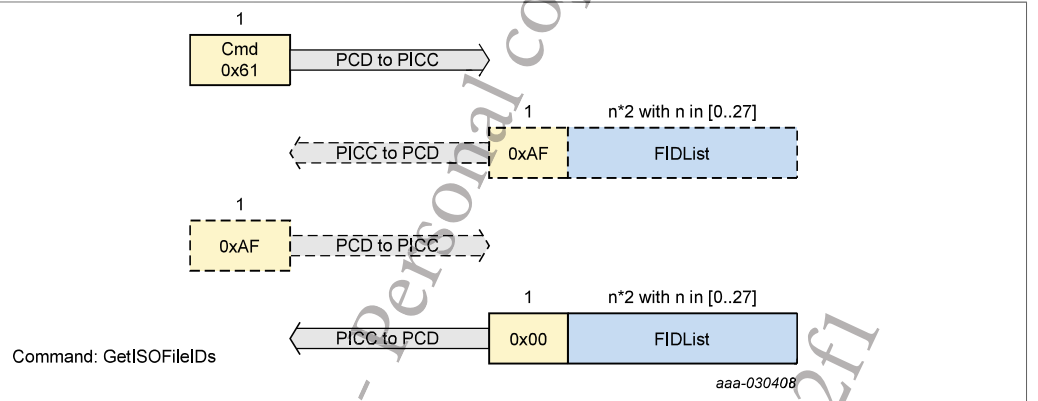


Figure 63. GetISOFileIDs command protocol

Table 228. Command description - GetISOFileIDs

GetISOFileIDs	
Description:	Get back the ISO File IDs
CommMode:	CommMode.MAC

Table 229. Command parameters description - GetISOFileIDs

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x61	Command code
Command Data Parameters			
-	-	-	No data parameters

Table 230. Return code description - GetISOFileIDs

Status	Value	Description
ADDITIONAL_FRAME	0xAF	Note that command chaining is only applied if the list does not fit into one response frame. In this case, the list is split between two ISO File IDs, i.e. never a partial ISO File ID is sent
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed
FILE_NOT_FOUND	0xF0	Application was created with ISO/IEC 7816-4 file identifiers disabled
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.AppMasterKey while required from AppKeySettings

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantas IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c2a4-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 230. Return code description - GetISOFileIDs*...continued*

Status	Value	Description
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting KeyID.AppMasterKey access rights while required from AppKeySettings

Table 231. Response data parameters description - GetISOFileIDs - ADDITIONAL_FRAME

Name	Length	Value	Description
FIDList	n × 2 with n in [0..27]	Full Range	List of n ISO File IDs

Table 232. Response data parameters description - GetISOFileIDs - OPERATION_OK

Name	Length	Value	Description
FIDList	n × 2 with n in [0..27]	Full Range	List of n ISO File IDs

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.7.10 GetFileSettings

The detailed description of this command can be found in Section 7.12.3.1.

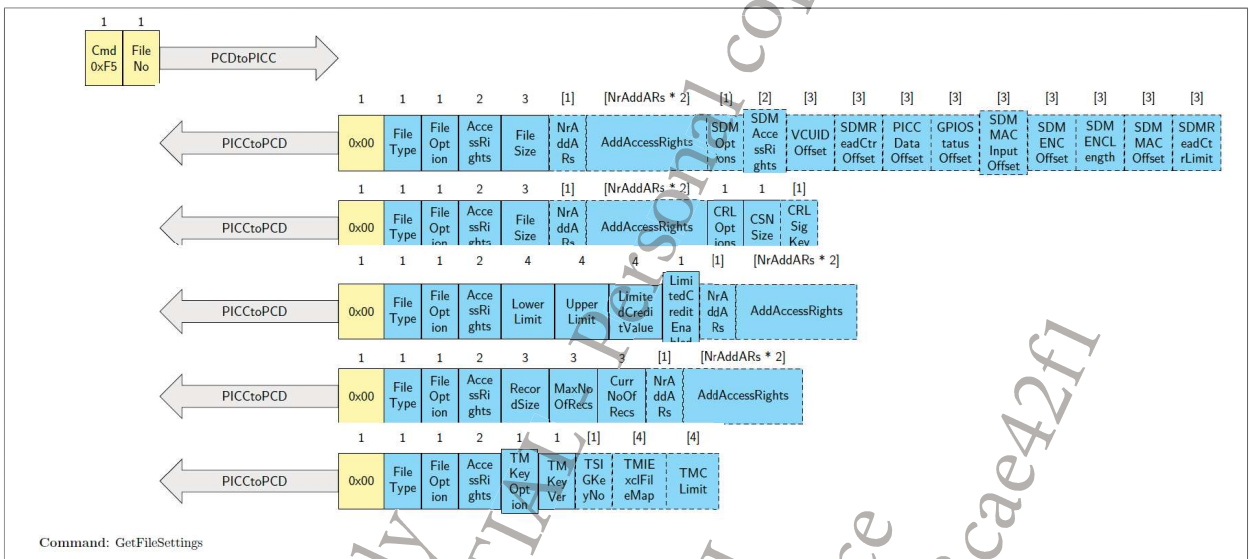


Figure 64. GetFileSettings command protocol

Table 233. Command description - GetFileSettings

GetFileSettings	
Description:	Get information on the properties of a specific file
CommMode:	CommMode.MAC

Table 234. Command parameters description - GetFileSettings

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xF5	Command code
FileNo	1	-	File number of the targeted file
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Limited range	
Command Data Parameters			
-	-	-	No data parameters

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 235. Return code description - GetFileSettings

Status	Value	Description
OPERATION_OK	0x00	Targeting FileType.StandardData or FileType.BackupData
		Targeting CRLFile
		Targeting FileType.Value
		Targeting FileType.TransactionMAC
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed.
PERMISSION_DENIED	0x9D	SAI given but no 2nd application selected
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.AppMasterKey while required from AppKeySettings
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting KeyID.AppMasterKey access rights while required from AppKeySettings
FILE_NOT_FOUND	0xF0	File with targeted FileNo does not exist for the targeted application.

Table 236. Response data parameters description - GetFileSettings - Targeting FileType.StandardData or FileType.BackupData

Name	Length	Value	Description
FileType	1	-	File Type of the targeted file
		0x00	FileType.StandardData
		0x01	FileType.BackupData
FileOption	1	-	Options for the targeted file
	Bit 7		Additional Access Rights
		0b	disabled
		1b	enabled
	Bit 6		Secure Dynamic Messaging and Mirroring
		0b	disabled
		1b	enabled
	Bit 5-4		Reserved
		00b	
	Bit 3-2		RFU
		00b	
	Bit 1-0		CommMode (see Table 7)
		-	
AccessRights	2	-	Set of access conditions for the 1st set in the file (see Table 41).

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 236. Response data parameters description - GetFileSettings - Targeting FileType.StandardData or FileType.BackupData...*continued*

Name	Length	Value	Description
FileSize	3	Limited range	File size of the targeted file
NrAddARs	[1]	0x00 .. 0x07	[Optional, present if Bit 7 of FileOption set] Number of additional access conditions sets (not including the 1st and mandatory one) of the targeted file
AddAccessRights	[NrAdd ARs × 2]	-	[Optional, present if Bit 7 of FileOption set] Sets of access conditions to update the 2nd up to (NrAddARs+1)th set of the file (see Table 41)
SDMOptions	[1]	-	[Optional, present if Bit 6 of FileOption set] SDM Options (see Table 246)
SDMAccessRights	[2]	-	[Optional, present if FileOption[b6] set] SDM Access Rights (see Table 246)
VCUIDOffset	[3]	-	[Optional, present if ((SDMOptions[b7] = 1) AND (FileAR.SDMMetaRead =0xE)] Mirror position (LSB first) for VCUID (see Table 246)
SDMReadCtrOffset	[3]	-	[Optional, present if ((SDMOptions[b6] = 1) AND (FileAR.SDMMetaRead =0xE)] Mirror position (LSB first) for SDMReadCtr (see Table 246)
PICCDataOffset	[3]	-	[Optional, present if FileAR.SDMMetaRead !=0xE and 0xF] Mirror position (LSB first) for encrypted PICCData (see Table 246)
GPIOStatusOffset	[3]	-	[Optional, present if (SDMOptions[b3] = 1)] Mirror position (LSB first) for GPIOStatus
SDMMACInputOffset	[3]	-	[Optional, present if FileAR.SDMFileRead != 0xF] Offset in the file where the SDM MAC computation starts (LSB first) (see Table 246)
SDMENCOffset	[3]	-	[Optional, present if ((FileAR.SDMFileRead != 0xF) AND (SDMOptions[b4] = 1))] SDMENCFileData mirror position (LSB first) (see Table 246)
SDMENCLength	[3]	-	[Optional, present if ((FileAR.SDMFileRead != 0xF) AND (SDMOptions[b4] = 1))] Length of the SDMENCFileData (LSB first) (see Table 246)
SDMMACOffset	[3]	-	[Optional, present if FileAR.SDMFileRead != 0xF] SDMMAC mirror position (LSB first) (see Table 246)
SDMReadCtrLimit	[3]	-	[Optional, present if SDMOptions[b5] = 1] SDMReadCtrLimit value (LSB first) (see Table 246)

Table 237. Response data parameters description - GetFileSettings - Targeting CRLFile

Name	Length	Value	Description
FileType	1	-	File Type of the targeted file
		0x00	FileType.StandardData
		0x01	FileType.BackupData
FileOption	1	-	Options for the targeted file
	Bit 7		Additional Access Rights

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07f9d-c22f-4f06-90fc-ec5ca42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 237. Response data parameters description - GetFileSettings - Targeting CRLFile...*continued*

Name	Length	Value	Description
		0b	disabled
		1b	enabled
	Bit 6		Secure Dynamic Messaging and Mirroring
		0b	disabled
	Bit 5-2		RFU
		0000b	
	Bit 1-0		CommMode (see Table 7)
		-	
AccessRights	2	-	Set of access conditions for the 1st set in the file (see Table 41)
FileSize	3	Limited range	File size of the targeted file
NrAddARs	[1]	0x00 .. 0x07	[Optional, present if Bit 7 of FileOption set] Number of addiitional access conditions sets (not including the 1st and mandatory one) of the targeted file.
AddAccessRights	[NrAddARs × 2]	-	[Optional, present if Bit 7 of FileOption set] Sets of access conditions to update the 2nd up to (NrAddARs+1)th set of the file (see Table 41)
CRLOptions	1	-	CRL Options (see Table 246)
CSNSize	1	-	CSN Size (see Table 246)
CRLSigKey	[1]	-	[Optional, present if CRLOptions[b0] is set] CRLSignature Key (see Table 246)

Table 238. Response data parameters description - GetFileSettings - Targeting FileType.Value

Name	Length	Value	Description
FileType	1	0x02	File Type of the targeted file
FileOption	1	-	Options for the targeted file
	Bit 7		Additional Access Rights
		0b	disabled
		1b	enabled
	Bit 6		RFU
		0b	
	Bit 5-4		Reserved
		00b	
	Bit 3-2		RFU
		00b	
	Bit 1-0		CommMode (see Table 7)
		-	
AccessRights	2	-	Set of access conditions for the 1st set in the file (see Table 41)

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 238. Response data parameters description - GetFileSettings - Targeting FileType.Value...*continued*

Name	Length	Value	Description
LowerLimit	4	Full Range	Lower Limit of the file (as defined at file creation)
UpperLimit	4	Full Range	Upper Limit of the file (as defined at file creation).
LimitedCreditValue	4	0x00000000 .. 0x7FFFFFFF	Current maximum Limited Credit Value (all zero if disabled)
LimitedCreditEnabled	1	-	Encodes if LimitedCredit is allowed for this file
	Bit 7-2		RFU
		000000b	
	Bit 1		Free GetValue
		0b	No free access to GetValue
		1b	Free access to GetValue
	Bit 0		LimitedCredit support
		0b	LimitedCredit disabled
		1b	LimitedCredit enabled
NrAddARs	[1]	0x00 .. 0x07	[Optional, present if Bit 7 of FileOption set] Number of addiitional access conditions sets (not including the 1st and mandatory one) of the targeted file
AddAccessRights	[NrAddARs × 2]	-	[Optional, present if Bit 7 of FileOption set] Sets of access conditions to update the 2nd up to (NrAddARs+1)th set of the file (see Table 41)

Table 239. Response data parameters description - GetFileSettings - Targeting FileType.LinearRecord or FileType.CyclicRecord

Name	Length	Value	Description
FileType	1	-	File Type of the targeted file.
		0x03	FileType.LinearRecord
		0x04	FileType.CyclicRecord
FileOption	1	-	Options for the targeted file
	Bit 7		Additional Access Rights
		0b	disabled
		1b	enabled
	Bit 6-2		RFU
		000000b	
	Bit 1-0		CommMode (see Table 7)
		-	
AccessRights	2	-	Set of access conditions for the 1st set in the file (see Table 41)
RecordSize	3	Limited Range	Size of a single record in the file (as defined at file creation)

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 239. Response data parameters description - GetFileSettings - Targeting FileType.LinearRecord or FileType.CyclicRecord...*continued*

Name	Length	Value	Description
MaxNoOfRecs	3	Limited Range	Maximum number of records within the file (as defined at file creation).
CurrNoOfRecs	3	-	Current number of records within the file
		0x00 .. Max NoOfRecs	FileType.LinearRecord
		0x00 .. (MaxNoOf Recs - 1)	FileType.CyclicRecord
NrAddARs	[1]	0x00 .. 0x07	[Optional, present if Bit 7 of FileOption set] Number of addiitional access conditions sets (not including the 1st and mandatory one) of the targeted file
AddAccessRights	[NrAdd ARs × 2]	-	[Optional, present if Bit 7 of FileOption set] Sets of access conditions to update the 2nd up to (NrAddARs+1)th set of the file (see Table 41)

Table 240. Response data parameters description - GetFileSettings - Targeting FileType.TransactionMAC

Name	Length	Value	Description
FileType	1	0x05	File Type of the targeted file
FileOption	1	-	Options for the targeted file
	Bit 7		RFU
		0b	
	Bit 6		TMI exclusion filemap
		0b	disabled
		1b	enabled
	Bit 5		TMCLimit configuration
		0b	disabled
		1b	enabled
	Bit 4		Reserved
		0b	
	Bit 3-2		RFU
		00b	
	Bit 1-0		CommMode (see Table 7)
		-	
AccessRights	2	-	Access conditions of the targeted file (see Table 40)
TMKeyOption	1	-	Options for KeyID.AppTransactionMACKey
	Bit 7-2		RFU
		000000b	
	Bit 7		Mode

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 240. Response data parameters description - GetFileSettings - Targeting FileType.TransactionMAC...*continued*

Name	Length	Value	Description
		0b	Transaction MAC
		1b	Transaction Signature
	Bit 6-2		RFU
		00000b	
	Bit 1-0		KeyType of the KeyID.AppTransactionMACKey
		00b	No KeyID.AppTransactionMACKey
		10b	KeyType.AES128
		11b	KeyType.AES256
TMKeyVer	1	Full Range	[Optional, present if configured KeyID.AppTransactionMACKey] Key version of KeyID.AppTransactionMACKey
TSIGKeyNo	[1]	-	[Optional, present if Transaction Signature] KeyID.AppTransactionSIGKey
	Bit 7-3		RFU
		00000b	
	Bit 2-0		KeyNo of configured KeyID.ECCPrivateKey
		0x0..0x04	
TMIExclFileMap	[4]	Full range	[Optional, present if Bit 6 of FileOption set] TMI exclusion filemap
TMCLimit	[4]	0x000000 01 .. 0xFF FFFFFF	[Optional, present if Bit 5 of FileOption set] TMCLimit

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ec3cae42f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.7.11 GetFileCounters

The detailed description of this command can be found in Section 7.12.3.2.

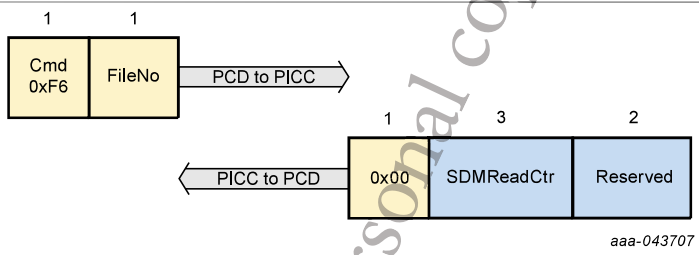


Figure 65. GetFileCounters command protocol

Table 241. Command description - GetFileCounters

GetFileCounters	
Description:	Get file related counters used for Secure Dynamic Messaging.
CommMode:	CommMode.Full

Table 242. Command parameters description - GetFileCounters

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xF6	Command code
FileNo	1	-	File number of the targeted file
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Limited range	
Command Data Parameters			
-	-	-	No data parameters

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90f6-5ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 243. Return code description - GetFileCounters

Status	Value	Description
OPERATION_OK	0x00	Targeting FileType.StandardData with SDM enabled
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		Targeted file is not of FileType.StandardData
		Targeted FileType.StandardData file has no Secure Dynamic Messaging enabled
		Targeted FileType.StandardData file has FileAR.SDMCtrRet set to 0xF
AUTHENTICATION_ERROR	0xAE	FileAR.SDMCtrRet not granted (while different from 0xF) for targeted FileType.StandardData file due to missing authentication or authentication with the wrong key
CERT_ERROR	0xCE	Active ECC-based authentication not granting FileAR.SDMCtrRet for targeted FileType.StandardData file
FILE_NOT_FOUND	0xF0	File with targeted FileNo does not exist for the targeted application.

Table 244. Response data parameters description - GetFileCounters - OPERATION_OK - Targeting FileType.StandardData with SDM enabled

Name	Length	Value	Description
SDMReadCtr	3	Full Range	Current SDMReadCtr of the targeted file (LSB first)
Reserved	2	0x0000	Reserved

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fc-6ec3ee42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.7.12 ChangeFileSettings

The detailed description of this command can be found in Section 7.12.2.3.

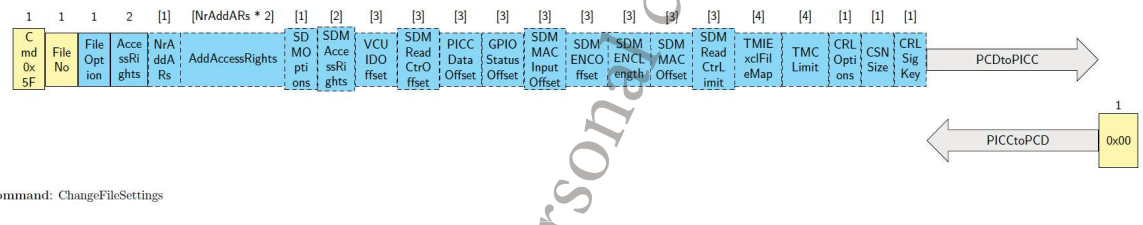


Figure 66. ChangeFileSettings command protocol

Table 245. Command description - ChangeFileSettings

ChangeFileSettings	
Description:	Changes the access parameters of an existing file
CommMode:	CommMode.Full

Table 246. Command parameters description - ChangeFileSettings

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x5F	Command code
FileNo	1	-	File number of the targeted file
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Limited range	
Command Data Parameters			
FileOption	1	-	Options for the targeted file
	Bit 7		[if not targeting FileType.TransactionMAC] Additional Access Rights
		0b	Not present
		1b	Present
	Bit 7		[if targeting FileType.TransactionMAC] RFU
		0b	
	Bit 6		[if targeting FileType.StandardData] Secure Dynamic Messaging and Mirroring
		0b	disabled

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Serafi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-022c-405c-90f6-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 246. Command parameters description - ChangeFileSettings...*continued*

Name	Length	Value	Description
		1b	enabled
	Bit 6		[if targeting FileType.TransactionMAC] TMI exclusion filemap
		0b	disabled
		1b	enabled
	Bit 6		[else] RFU
		0b	
	Bit 5		[if targeting FileType.TransactionMAC] TMCLimit configuration
		0b	disabled
		1b	enabled
	Bit 5		[else] RFU
		0b	
	Bit 4		Reserved
		0b	
	Bit 3-2		RFU
		00b	
	Bit 1-0		CommMode (see Table Table 7)
		-	
AccessRights	2	-	Set of access conditions for the 1st set in the file.
NrAddARs	[1]	0x00 .. 0x07	[Optional, present if Bit 7 of FileOption set] Number of addiitional access conditions sets (not including the 1st and mandatory one) of the targeted file
AddAccessRights	[NrAddARs × 2]	Limited range	[Optional, present if Bit 7 of FileOption set] Sets of access conditions to update the 2nd up to (NrAddARs+1th) set of the file (see Table 41)
SDMOptions	[1]	-	[Optional, present if FileOption[b6] set and targeting FileType.StandardData] SDM Options
	Bit 7		VCUID (only for mirroring)
		0b	disabled
		1b	enabled
	Bit 6		SDMReadCtr
		0b	disabled
		1b	enabled
	Bit 5		SDMReadCtrLimit
		0b	disabled

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 246. Command parameters description - ChangeFileSettings...*continued*

Name	Length	Value	Description
		1b	enabled
	Bit 4		SDMENCFileData
		0b	disabled
		1b	enabled
	Bit 3		GPIOStatus
		0b	disabled
		1b	enabled
	Bit 2-1		RFU
		00b	
	Bit 0		Encoding mode
		0b	Reserved
		1b	ASCII
SDMAccessRights	[2]	-	[Optional, present if FileOption[b6] set and targeting FileType.StandardData] SDM Access Rights
	Bit 15-12		FileAR.SDMMetaRead
		0x0..0xD	Encrypted PICCData mirroring using the targeted KeyID.AppKey (max key number depends on number of application keys)
		0xE	Plain PICCData mirroring
		0xF	No PICCData mirroring
	Bit 11-8		FileAR.SDMFileRead
		0x0..0xD	Targeted KeyID.AppKey (max key number depends on number of application keys)
		0xF	No symmetric SDM for Reading
	Bit 7-4		FileAR.SDMFileRead2
		0x0..0x4	Targeted KeyID.ECCPrivateKey
		0xF	No asymmetric SDM for Reading
	Bit 3-0		FileAR.SDMCtrRet
		0x0..0xD	Targeted KeyID.AppKey
		0xE	Free
		0xF	No Access
VCUIDOffset	[3]	-	[Optional, present if ((SDMOptions[b7] = 1) AND (FileAR.SDMMetaRead =0xE)] Mirror position (LSB first) for VCUID
		0x0 .. (FileSize - VCUID-Length)	Offset within the file

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-2a4-405f-9ate-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 246. Command parameters description - ChangeFileSettings...*continued*

Name	Length	Value	Description
SDMReadCtrOffset	[3]	-	[Optional, present if ((SDMOptions[b6] = 1) AND (FileAR.SDMMetaRead =0xE)] Mirror position (LSB first) for SDMReadCtr
		0x0 .. (FileSize - SDMReadCtrLength)	Offset within the file
		0xFFFFFF	No SDMReadCtr mirroring
PICCDataOffset	[3]	-	[Optional, present if FileAR.SDMMetaRead !=0xE and 0xF] Mirror position (LSB first) for encrypted PICCData
		0x0 .. (FileSize - PICCData-Length)	Offset within the file
GPIOStatusOffset	[3]	-	[Optional, present if (SDMOptions[b3] = 1)] Mirror position (LSB first) for GPIOStatus .
		0x0 .. (FileSize-3)	Offset within the file
SDMMACInputOffset	[3]	-	[Optional, present if (FileAR.SDMFileRead != 0xF) OR (FileAR.SDMFileRead2 != 0xF)] Offset in the file where the SDM MAC or Signature computation starts (LSB first)
		0x0 .. (SDMMAC-Offset)	Offset within the file
SDMENCOffset	[3]	-	[Optional, present if ((FileAR.SDMFileRead != 0xF) AND (SDMOptions[b4] = 1))] SDMENCFileData mirror position (LSB first)
		SDMMAC-InputOffset .. (SDMMAC-Offset - 32)	[if (SDMMACOffset != 0xFFFFFE) AND ASCII encoding] Offset within the file
SDMENCLength	[3]	-	[Optional, present if ((FileAR.SDMFileRead != 0xF) AND (SDMOptions[b4] = 1))] Length of the SDMENCFileData (LSB first)
		32 .. (SDMMAC-Offset - SDMENC-Offset)	[if ASCII encoding] Offset within the file, must be multiple of 32
SDMMACOffset	[3]	-	[Optional, present if (FileAR.SDMFileRead != 0xF) OR (FileAR.SDMFileRead2 != 0xF)] SDMMAC or SDMSIG mirror position (LSB first)
		SDMMAC-InputOffset .. (FileSize - 16)	[if (FileAR.SDMFileRead != 0xF) AND (FileAR.SDMFileRead2 == 0xF) AND (SDMOptions[b4] = 0) AND ASCII encoding]
		SDMMAC-InputOffset .. (FileSize - 128)	[if (FileAR.SDMFileRead2 != 0xF) AND (SDMOptions[b4] = 0) AND ASCII encoding
		(SDMENC-Offset + SDMENC-Length) .. (FileSize-16)	[if (FileAR.SDMFileRead != 0xF) AND (FileAR.SDMFileRead2 == 0xF) AND (SDMOptions[b4] = 1) AND ASCII encoding]

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions CorbEA
Do not copy, share or reproduce
5fc779d-c2c4-455-901e-5ecc5ee4af1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 246. Command parameters description - ChangeFileSettings...*continued*

Name	Length	Value	Description
		(SDMENC-Offset + SDMENC-Length) .. (FileSize-128)	[if (FileAR.SDMFileRead2 != 0xF) AND (SDMOptions[b4] = 1) AND ASCII encoding]
SDMReadCtrLimit	[3]	Full range	[Optional, present if SDMOptions[b5] = 1] SDMReadCtrLimit value (LSB first)
TMIExclFileMap	[4]	Full range	[Optional, present if FileOption[b6] set and targeting FileType.TransactionMAC] TMI exclusion filemap
TMCLimit	[4]	TMC+1.. 0xFFFFFFFF	[Optional, present if FileOption[b5] = 1] TMCLimit
CRLOptions	[1]	-	[Optional, present if targeting CRLFile] CRL Options
	Bit 7-2		RFU
		000000b	
	Bit 1		CRLVersion required to be incremented by 1
		0b	disabled
		1b	enabled
	Bit 0		CRLSignature and monotonic CRLVersion required.
		0b	disabled
		1b	enabled
CSNSize	[1]	-	[Optional, present if targeting CRLFile] CSN Size
		0x00	Certificate revocation disabled
		0x04	CSN Size of 4 byte
CRLSigKey	[1]	-	[Optional, present if targeting CRLFile and CRLOptions[b0] is set] CRLSignature Key
		0x0..0x1	[if AID 0x000000 selected] At PICC level, two KeyID.CARootKey s are supported
		0x0..0x4	[if AID 0x000000 is not selected] At application level, up to five KeyID.CARootKeys are supported

Table 247. Return code description - ChangeFileSettings

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid cryptogram (padding or CRC). Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae4211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 247. Return code description - ChangeFileSettings...*continued*

Status	Value	Description
PARAMETER_ERROR	0x9E	Parameter value not allowed
		Targeted key for FileAR.SDMMetaRead or FileAR.SDMFileRead is not an existing symmetric key
		Targeted KeyID.ECCPrivateKey for FileAR.SDMFileRead2 is not existing or not enabled for ECC-based SDM
		Trying to set FileAR.SDMMetaRead to a value different than 0xF, while both VCUID and SDMReadCtr mirroring are disabled
		Trying to set FileAR.SDMMetaRead to 0xF, while enabling VCUID mirroring
		Trying to set FileAR.SDMCtrRet to a value different from 0xF, while SDMReadCtr is disabled
		SDMMAC and VCUID mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset >= VCUIDOffset + VCUIDLength) OR (VCUIDOffset >= SDMMACOffset + SDMMACLength)
		SDMMAC and SDMReadCtr mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset >= SDMReadCtrOffset + SDMReadCtrLength OR (SDMReadCtrOffset >= SDMMACOffset + SDMMACLength)
		SDMMAC and PICCData mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset >= PICCDataOffset + PICCDataLength) OR (PICCDataOffset >= SDMMACOffset + SDMMACLength)
		SDMMAC and GPIOStatus mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset >= GPIOStatus + 3) OR (GPIOStatus >= SDMMACOffset + SDMMACLength)
		SDMSIG and GPIOStatus mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset >= GPIOStatus + 3) OR (GPIOStatus >= SDMMACOffset + SDMSIGLength)
		SDMSIG and VCUID mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset >= VCUIDOffset + VCUIDLength) OR (VCUIDOffset >= SDMMACOffset + SDMSIGLength)

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or replace
5fc0779d-520-40fc-90fe-66c23aea21

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 247. Return code description - ChangeFileSettings...*continued*

Status	Value	Description
		SDMSIG and SDMReadCtr mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset >= SDMReadCtrOffset + SDMReadCtrLength) OR (SDMReadCtrOffset >= SDMMACOffset + SDMSIGLength)
		SDMSIG and PICCData mirroring are overlapping, i.e. the following condition is not satisfied: (SDMMACOffset >= PICCDataOffset + PICCDataLength) OR (PICCDataOffset >= SDMMACOffset + SDMSIGLength)
		SDMENCFileData and VCUID mirroring are overlapping, i.e. the following condition is not satisfied: (SDMENCOffset >= VCUIDOffset + VCUIDLength) OR (VCUIDOffset >= SDMENCOffset + SDMENCLength)
		SDMENCFileData and SDMReadCtr mirroring are overlapping, i.e. the following condition is not satisfied: (SDMENCOffset >= SDMReadCtrOffset + SDMReadCtrLength) OR (SDMReadCtrOffset >= SDMENCOffset + SDMENCLength)
		SDMENCFileData and PICCData mirroring are overlapping, i.e. the following condition is not satisfied: (SDMENCOffset >= PICCDataOffset + PICCDataLength) OR (PICCDataOffset >= SDMENCOffset + SDMENCLength)
		GPIOStatus and VCUID mirroring are overlapping, i.e. the following condition is not satisfied: (GPIOStatus >= VCUIDOffset + VCUIDLength) OR (VCUIDOffset >= GPIOStatus + 3)
		GPIOStatus and SDMReadCtr mirroring are overlapping, i.e. the following condition is not satisfied: (GPIOStatus >= SDMReadCtrOffset + SDMReadCtrLength) OR (SDMReadCtrOffset >= GPIOStatus + 3)

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 247. Return code description - ChangeFileSettings...*continued*

Status	Value	Description
		GPIOStatus and PICCData mirroring are overlapping, i.e. the following condition is not satisfied: (GPIOStatus >= PICCDataOffset + PICCDataLength) OR (PICCDataOffset >= GPIOStatus + 3)
		GPIOStatus is overlapping with SDMENCFileData without being fully part of the plain input data area, i.e. following condition is not satisfied: (GPIOStatus + 3 <= SDMENCOffset) OR (GPIOStatus >= SDMENCOffset + SDMENCLength) OR (GPIOStatus >= SDMENCOffset) AND ((GPIOStatus + 3) <= (SDMENCOffset + SDMENCLength /2))
		VCUID and SDMReadCtr mirorring are overlapping, i.e. the following condition is not satisfied: (VCUIDOffset >= SDMReadCtrOffset + SDMReadCtrLength) OR (SDMReadCtrOffset >= VCUIDOffset + VCUIDLength)
		Enabling Secure Dynamic Messaging encryption (SDMOptions[b4] set to 1) is not possible if FileAR.SDMFileRead = 0xF.
		Enabling Secure Dynamic Messaging encryption (SDMOptions[b4] set to 1) is not allowed if not both SDMReadCtr and VCUID are mirrored (i.e. SDMOptions[b7] and SDMOptions[b6] must be set to 1)
		Trying to set a TMCLimit which is smaller or equal to the current TMC
		Trying to set a SDMReadCtrLimit while not enabling SDMReadCtr
		Trying to set a SDMReadCtrLimit which is smaller or equal to the current SDMReadCtr
		Trying to set CRLOptions Bit 1 while Bit 0 is not set.
PERMISSION_DENIED	0x9D	FileAR.Change of targeted file has access conditions set to 0xF.
		Number of access condition per access right has been restricted to 1 at file creation and the command is trying to add access conditions.
		SAI given but no 2nd application selected.
		Enabling Secure Dynamic Messaging or TMI exclusion filemap (FileOption Bit 6 set to 1) is only allowed if targeting FileType.StandardData or FileType.TransactionMAC
		Enabling Secure Dynamic Messaging (FileOption Bit 6 set to 1) is not allowed if not pre-enabled on file creation of the targeted file

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-02c-4a5c-90fe-6ec3cad211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 247. Return code description - ChangeFileSettings*...continued*

Status	Value	Description
		Trying to enable GPIOStatus while GPIO support disabled by product configuration
		Trying to enable CommitReaderID (i.e. FileAR{ReadWrite} = 0x0..0xE) without available KeyID.AppTransactionMACKey
FILE_NOT_FOUND	0xF0	File with targeted FileNo does not exist for the targeted application.
AUTHENTICATION_ERROR	0xAE	FileAR.Change of targeted file not granted as there is no active authentication with the required key while the access conditions is different from 0xF
CERT_ERROR	0xCE	Active ECC-based authentication not granting FileAR.Change access rights

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.8 Data Management

8.8.1 ReadData

The detailed description of this command can be found in Section 7.13.1.1.

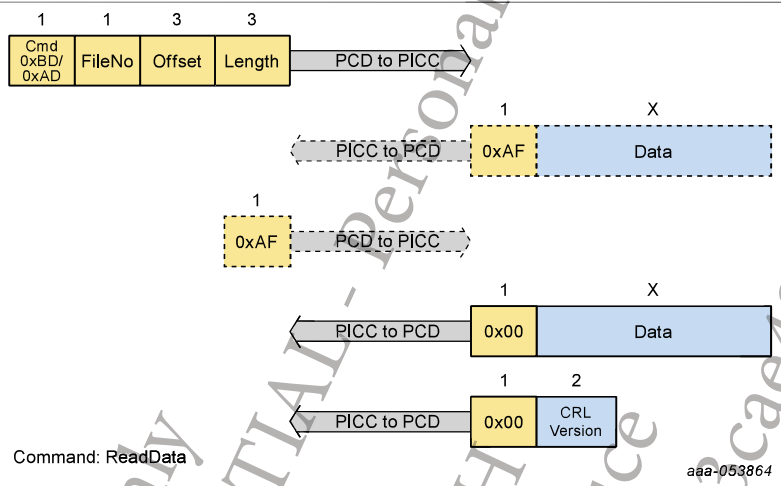


Figure 67. ReadData command protocol

Table 248. Command description - ReadData

ReadData	
Description:	Reads data from FileType.StandardData, FileType.BackupData or FileType.TransactionMAC files.
CommMode:	CommMode of targeted file

Table 249. Command parameters description - ReadData

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xBD/0xAD	Command code. 0xBD supports native chaining, while 0xAD supports larger FSC and ISO/IEC 14443-4 chaining
FileNo	1	-	File number of the targeted file
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Limited range	
Offset	3	-	Offset information
		0x000000 .. (FileSize - 1)	Starting position for the read operation

Provided under NDA only
COMPANY CONFIDENTIAL - personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 249. Command parameters description - ReadData...*continued*

Name	Length	Value	Description
		0xFFFFFF	Return CRLFile meta-data
Length	3	-	Number of bytes to be read
		0x000000	Read the entire data file, starting from the position specified in the offset value
		0x000001 .. (FileSize - Offset)	Read the given amount of bytes
Command Data Parameters			
-	-	-	No data parameters

Table 250. Return code description - ReadData

Status	Value	Description
ADDITIONAL_FRAME	0xAF	Only for 0xBD variant
OPERATION_OK	0x00	
		Targeting CRLFile with Offset = 0xFFFFFF
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
		SDMReadCtr overflow
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	Targeted file is not of FileType.StandardData, FileType.BackupData or FileType.TransactionMAC
		FileAR.Read and FileAR.ReadWrite of targeted FileType.StandardData or FileType.BackupData file only have access conditions set to 0xF
		FileAR.Read of targeted FileType.TransactionMAC file only have access conditions set to 0xF
		FileAR.Read, FileAR.ReadWrite and FileAR.SDMFileRead (if SDM enabled) of targeted FileType.StandardData file only have access conditions set to 0xF
		Targeted file cannot be read in VCState.NotAuthenticated as the related SDMReadCtr is equal or bigger than its SDMReadCtrLimit
		SAI given but no 2nd application selected
		Trying to read SDMSIG while the KeyUsageCtrLimit of the targeted key entry is enabled and reached
		Trying to read SDMSIG while the key configured with FileAR.SDMFileRead2 does not exist or is not enabled for SDM
FILE_NOT_FOUND	0xF0	Targeted file does not exist in the targeted application
APPLICATION_NOT_FOUND	0xA0	Wrong application selected after NFCPause release

Provided under NDA only
COMPANY CONFIDENTIAL - Persona copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-0a2-0a2-0a2-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 250. Return code description - ReadData...*continued*

Status	Value	Description
AUTHENTICATION_ERROR	0xAE	FileAR.Read of targeted FileType.TransactionMAC file not granted while different from 0xF
		FileAR.Read , FileAR.ReadWrite and FileAR.SDMFileRead (if SDM enabled) of targeted file not granted while at least one of the access conditions is different from 0xF
CERT_ERROR	0xCE	Active ECC-based authentication not granting the required access rights
BOUNDARY_ERROR	0xBE	If targeting FileType.StandardData or FileType.BackupData, attempt to read beyond the file boundary as set during creation
		If targeting FileType.TransactionMAC , attempt to read beyond the file boundary of 12 byte

Table 251. Response data parameters description - ReadData - ADDITIONAL_FRAME

Name	Length	Value	Description
Data	X	Full range	Data read. The number of bytes to send to the PCD does not fit into one single frame, the PICC waits for a status frame with status byte 0xAF, before the next frame is sent to the PCD

Table 252. Response data parameters description - ReadData - OPERATION_OK

Name	Length	Value	Description
Data	X	Full range	Data read

Table 253. Response data parameters description - ReadData - OPERATION_OK - Targeting CRLFile with Offset = 0xFFFFFF

Name	Length	Value	Description
CRLVersion	2	-	CRLVersion (see Table 255)

Provided under NDA only - Personal copy for:
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc5ae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.8.2 WriteData

The detailed description of this command can be found in Section 7.13.1.2.

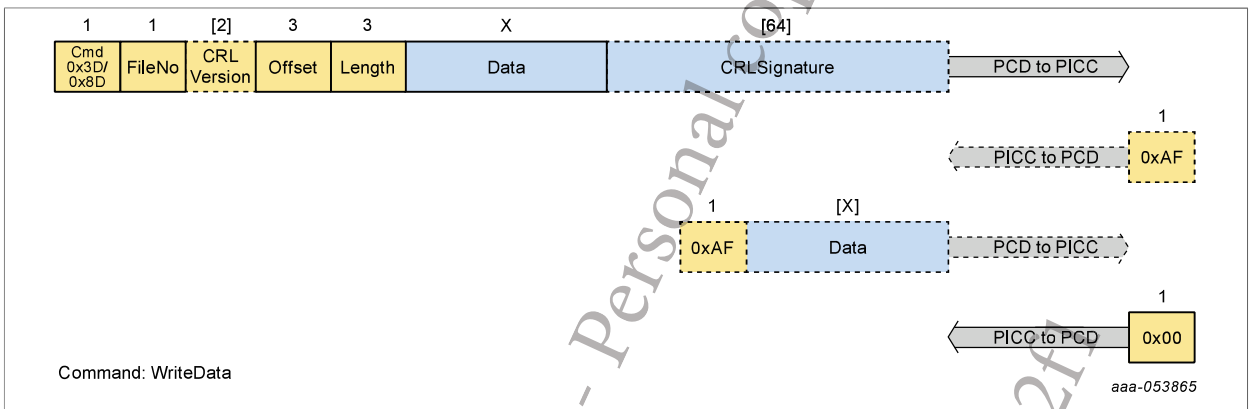


Figure 68. WriteData command protocol

Table 254. Command description - WriteData

WriteData	
Description:	Writes data to FileType.StandardData and FileType.BackupData files.
CommMode:	CommMode of targeted file

Table 255. Command parameters description - WriteData

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x3D/0x8D	Command code, 0x3D supports native chaining, while 0x8D supports larger FSC and ISO/IEC 14443-4 chaining
FileNo	1	-	File number of the targeted file
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Full Range	
CRLVersion	[2]	-	[Optional, present if targeting CRLFile] CRL Options
		Full range	[if CRLOptions.Bit0 == '0'] any CRLVersion
		Limited range	[if CRLOptions.Bit0 == '1' AND CRLOptions.Bit 1 == '0'] CRLVersion > current CRLVersion
		Limited range	[if CRLOptions.Bit0 == '1' AND CRLOptions.Bit 1 == '1'] CRLVersion == current CRLVersion + 1
Offset	3	0x000000 .. (FileSize - 1)	Starting position for the write operation

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 255. Command parameters description - WriteData...*continued*

Name	Length	Value	Description
Length	3	0x000001 .. (FileSize - Offset)	Number of bytes to be written
Command Data Parameters			
Data	X	Limited range	Data to be written
CRLSignature	[64]	Full range	[Optional, present if targeting CRLFile AND CRLOptions.Bit0=='1'] CRL Signature

Table 256. Return code description - WriteData

Status	Value	Description
ADDITIONAL_FRAME	0xAF	Only for 0x3D variant: If the number of bytes to send does not fit into one single frame, the PCD has to wait for a status frame with status byte 0xAF before sending the next frame to the PICC
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC or encryption padding
		Targeting CRLFile with invalid CRLSignature, while required by CRLOptions, Bit 0
		Targeting CRLFile which has not been associated with a valid KeyID.CARootKey (reference) via CRLSigKey, while CRLSignature is required by CRLOptions, Bit 0
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	Targeted file is not of FileType.StandardData or FileType.BackupData
		FileAR.Write and FileAR.ReadWrite of targeted file only have access conditions set to 0xF
		SAI given but no 2nd application selected
		Targeting a FileType.StandardData file with a chained command in CommMode.MAC or CommMode.Full in VCState.AuthenticatedAES while this is not allowed
		Targeting a FileType.StandardData file with a chained command in CommMode.MAC or CommMode.Full in VCState.AuthenticatedECC while this is not allowed
		Targeting a FileType.StandardData file with a chained command exceeding the supported buffer size while targeting a CRLFile
		Targeting a CRLFile with CRLVersion smaller or equal than the current CRLVersion, while CRLOptions.Bit0=`1'
		Targeting a CRLFile with CRLVersion different from current CRLVersion+1, while CRLOptions.Bit1=`1'
FILE_NOT_FOUND	0xF0	Targeted file does not exist in the targeted application.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-022A0bc-4cfc-acc9ae4241

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 256. Return code description - WriteData...*continued*

Status	Value	Description
AUTHENTICATION_ERROR	0xAE	FileAR.Write and FileAR.ReadWrite of targeted file not granted while at least one of the access conditions is different from 0xF
CERT_ERROR	0xCE	Active ECC-based authentication not granting the required access rights
BOUNDARY_ERROR	0xBE	Attempt to write beyond the file boundary as set during creation

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.8.3 GetValue

The detailed description of this command can be found in Section 7.13.2.1.

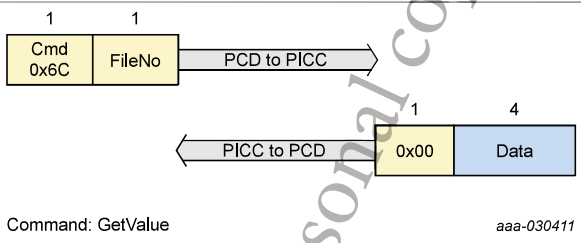


Figure 69. GetValue command protocol

Table 257. Command description - GetValue

GetValue	
Description:	Reads the currently stored value from a FileType.Value file.
CommMode:	CommMode of targeted file

Table 258. Command parameters description - GetValue

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x6C	Command code
FileNo	1	-	File number of the targeted file
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Full Range	
Command Data Parameters			
-	-	-	No data parameters

Table 259. Return code description - GetValue

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed

Provided under NDA only
COMPANY CONFIDENTIAL
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-ecc3cae42f1
Personal copy for:

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 259. Return code description - GetValue*...continued*

Status	Value	Description
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		Targeted file is not of FileType.Value
		FileAR.Read , FileAR.Write and FileAR.ReadWrite of targeted file only have access conditions set to 0xF
		SAI given but no 2nd application selected
FILE_NOT_FOUND	0xF0	Targeted file does not exist in the targeted application
AUTHENTICATION_ERROR	0xAE	FileAR.Read , FileAR.Write and FileAR.ReadWrite of targeted file not granted while at least one of the access conditions is different from 0xF and no free access was granted at file creation
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting the required access rights

Table 260. Response data parameters description - GetValue - OPERATION_OK

Name	Length	Value	Description
Data	4	Full Range	Current Value

Provided under NDA only - Personal copy for:
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae4251

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.8.4 Credit

The detailed description of this command can be found in Section 7.13.2.2.

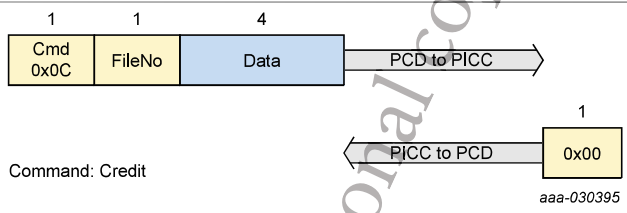


Figure 70. Credit command protocol

Table 261. Command description - Credit

Credit	
Description:	Increases a value stored in a FileType.Value file.
CommMode:	CommMode of targeted file

Table 262. Command parameters description - Credit

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x0C	Command code
FileNo	1	-	File number of the targeted file
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Full Range	
Command Data Parameters			
Data	4	0x00000000 .. 0x7FFFFFFF	Value to be credited. LSB first. Shall be positive

Table 263. Return code description - Credit

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC or encryption padding
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 263. Return code description - Credit...*continued*

Status	Value	Description
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		Targeted file is not of FileType.Value
		FileAR.ReadWrite of targeted file only have access conditions set to 0xF
		SAI given but no 2nd application selected
FILE_NOT_FOUND	0xF0	Targeted file does not exist in the targeted application
AUTHENTICATION_ERROR	0xAE	FileAR.ReadWrite of targeted file not granted while at least one of the access conditions is different from 0xF
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting the required access rights
BOUNDARY_ERROR	0xBE	Data added to the current value of the value file results in a value outside the LowerLimit and UpperLimit, or creates an integer arithmetic overflow

Provided under NDA only
COMPANY CONFIDENTIAL - Perceta copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae4ef1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.8.5 Debit

The detailed description of this command can be found in Section 7.13.2.3.

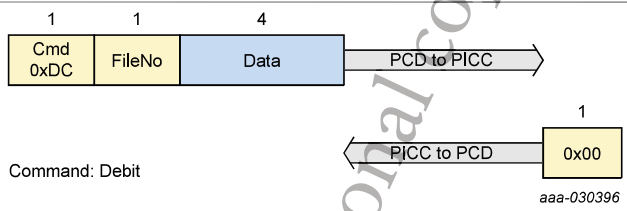


Figure 71. Debit command protocol

Table 264. Command description - Debit

Debit	
Description:	Decreases a value stored in a FileType.Value file.
CommMode:	CommMode of targeted file

Table 265. Command parameters description - Debit

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xDC	Command code
FileNo	1	-	File number of the targeted file
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Full Range	
Command Data Parameters			
Data	4	0x00000000 .. 0x7FFFFFFF	Value to be dedited. LSB first. Shall be positive.

Table 266. Return code description - Debit

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC or encryption padding
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 266. Return code description - Debit...*continued*

Status	Value	Description
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		Targeted file is not of FileType.Value .
		FileAR.Read, FileAR.Write and FileAR.ReadWrite of targeted file only have access conditions set to 0xF
		SAI given but no 2nd application selected
FILE_NOT_FOUND	0xF0	Targeted file does not exist in the targeted application
AUTHENTICATION_ERROR	0xAE	FileAR.Read, FileAR.Write and FileAR.ReadWrite of targeted file not granted while at least one of the access conditions is different from 0xF
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting the required access rights
BOUNDARY_ERROR	0xBE	Data subtracted from the current value of the value file results in a value outside the LowerLimit and UpperLimit, or creates an integer arithmetic underflow

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.8.6 LimitedCredit

The detailed description of this command can be found in Section 7.13.2.4.

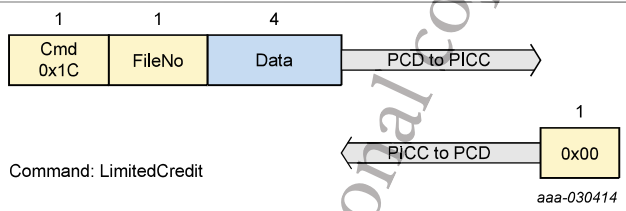


Figure 72. LimitedCredit command protocol

Table 267. Command description - LimitedCredit

LimitedCredit	
Description:	Allows a limited increase of a value stored in a FileType.Value file without having full Credit permissions to the file.
CommMode:	CommMode of targeted file

Table 268. Command parameters description - LimitedCredit

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x1C	Command code
FileNo	1	-	File number of the targeted file
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Full Range	
Command Data Parameters			
Data	4	0x00000000 .. 0x7FFFFFFF	Value to be credited, LSB first, Shall be positive

Table 269. Return code description - LimitedCredit

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC or encryption padding
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-ecc3cae4261

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 269. Return code description - LimitedCredit...*continued*

Status	Value	Description
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		Targeted file is not of FileType.Value
		FileAR.Write and FileAR.ReadWrite of targeted file only have access conditions set to 0xF
		SAI given but no 2nd application selected
		Limited credit not enabled for file with FileNo
		SAI given but no 2nd application selected
FILE_NOT_FOUND	0xF0	Targeted file does not exist in the targeted application
AUTHENTICATION_ERROR	0xAE	FileAR.Write and FileAR.ReadWrite of targeted file not granted while at least one of the access conditions is different from 0xF
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting the required access rights
BOUNDARY_ERROR	0xBE	Data is exceeding the limited credit limit
		Already successfully executed a LimitedCredit during the current ongoing transaction
		Already successfully executed and committed a LimitedCredit since the latest successful transaction containing at least one debit
		Data added to the current value of the value file results in a value outside the LowerLimit and UpperLimit, or creates an integer arithmetic overflow

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.8.7 ReadRecords

The detailed description of this command can be found in Section 7.13.3.1.

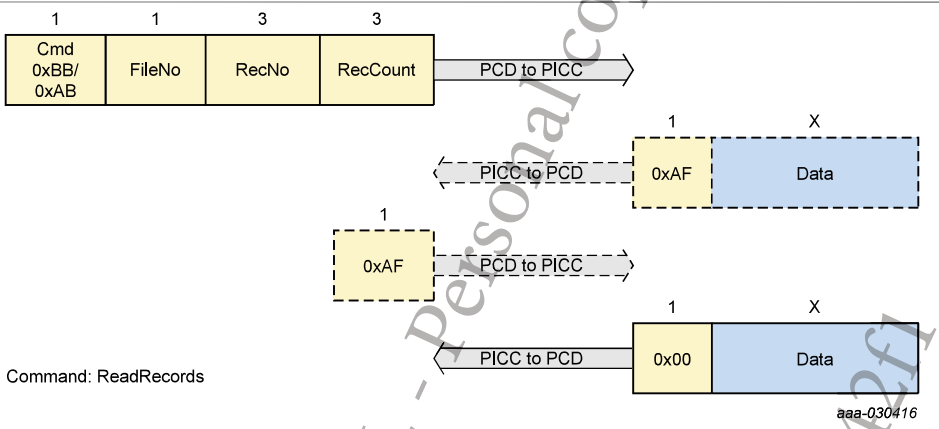


Figure 73. ReadRecords command protocol

Table 270. Command description - ReadRecords

ReadRecords	
Description:	Reads out a set of complete records from a FileType.LinearRecord or FileType.CyclicRecord File.
CommMode:	CommMode of targeted file

Table 271. Command parameters description - ReadRecords

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xBB/0xAB	Command code. 0xBB supports native chaining, while 0xAB supports larger FSC and ISO/IEC 14443-4 chaining
FileNo	1	-	File number of the targeted file
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Full Range	
RecNo	3	-	Record number. Records are numbered starting with the latest record written
		0x000000 .. (ExistingRecCount - 1)	0x000000 meaning the latest record written. The number is limited by the number of existing records (ExistingRecCount)

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-40bc-90fe-6ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 271. Command parameters description - ReadRecords...*continued*

Name	Length	Value	Description
RecCount	3	-	Number of records to be read. Records are always transmitted by the PICC in chronological order (= starting with the oldest, which is RecCount -1 before the one addressed by the given RecNo)
		0x000000	Read all records, from the oldest record up to and including the newest record (given by RecNo) are read
		0x000001 .. (ExistingRecCount - RecNo)	The number is limited by the number of existing records (ExistingRecCount) and the offset given by RecNo
Command Data Parameters			
-	-	-	No data parameters:Return code description - Read Records

Table 272. Return code description - ReadRecords

Status	Value	Description
ADDITIONAL_FRAME	0xAF	Only for 0xBB variant
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		Targeted file is not of FileType.LinearRecord or FileType.CyclicRecord
		FileAR.Read and FileAR.ReadWrite of targeted file only have access conditions set to 0xF
		SAI given but no 2nd application selected
FILE_NOT_FOUND	0xF0	Targeted file does not exist in the targeted application
AUTHENTICATION_ERROR	0xAE	FileAR.Read and FileAR.ReadWrite of targeted file not granted while at least one of the access conditions is different from 0xF
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting the required access rights
BOUNDARY_ERROR	0xBE	Attempt to read more records than have been written so far

Provided under NDA only
COMPANY CONFIDENTIAL - Personal Copy for:
Sergii Salata
Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-0e2c-40c9-9ee0-ec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 273. Response data parameters description - ReadRecords - ADDITIONAL_FRAME

Name	Length	Value	Description
Data	X	-	Data read. The number of bytes to send to the PCD does not fit into one single frame, the PICC waits for a status frame with status byte 0xAF, before the next frame is sent to the PCD
		Full Range	
		Full Range	

Table 274. Response data parameters description - ReadRecords - OPERATION_OK

Name	Length	Value	Description
Data	X	Full Range	Data read

Provided under NDA only
COMPANY CONFIDENTIAL - personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.8.8 WriteRecord

The detailed description of this command can be found in Section 7.13.3.2.

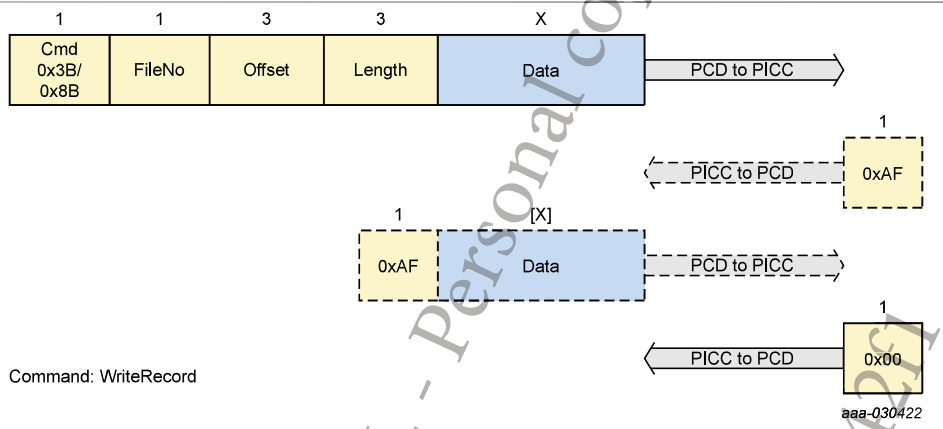


Figure 74. WriteRecord command protocol

Table 275. Command description - WriteRecord

WriteRecord	
Description:	Writes data to a record in a FileType.LinearRecord or FileType.CyclicRecord file.
CommMode:	CommMode of targeted file

Table 276. Command parameters description - WriteRecord

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x3B/0x8B	Command code. 0x3B supports native chaining, while 0x8B supports larger FSC and ISO/IEC 14443-4 chaining
FileNo	1	-	File number of the targeted file
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Full Range	
Offset	3	0x000000 .. (RecordSize - 1)	Starting position for the write operationwithin the targeted record
Length	3	0x000001 .. (RecordSize - Offset)	Number of bytes to be written
Command Data Parameters			
Data	X	Full Range	Data to be written

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 277. Return code description - WriteRecord

Status	Value	Description
ADDITIONAL_FRAME	0xAF	Only for 0x3B variant: If the number of bytes to send does not fit into one single frame, the PCD has to wait for a status frame with status byte 0xAF before sending the next frame to the PICC
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC or encryption padding
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		Targeted file is not of FileType.LinearRecord or FileType.CyclicRecord
		FileAR.Write and FileAR.ReadWrite of targeted file only have access conditions set to 0xF
		A UpdateRecord targeting the currently targeted file has already been performed within the current transaction
		A ClearRecordFile targeting the currently targeted file has already been performed within the current transaction
		SAI given but no 2nd application selected
FILE_NOT_FOUND	0xF0	Targeted file does not exist in the targeted application
AUTHENTICATION_ERROR	0xAE	FileAR.Write and FileAR.ReadWrite of targeted file not granted while at least one of the access conditions is different from 0xF
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting the required access rights
BOUNDARY_ERROR	0xBE	Offset + 1 is strictly bigger than the record size of the file
		Offset + Length + 1 is strictly bigger than the record size of the file
		Targeted file is of FileType.LinearRecord which is full, i.e. no new records can be written anymore

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.8.9 UpdateRecord

The detailed description of this command can be found in Section 7.13.3.3.

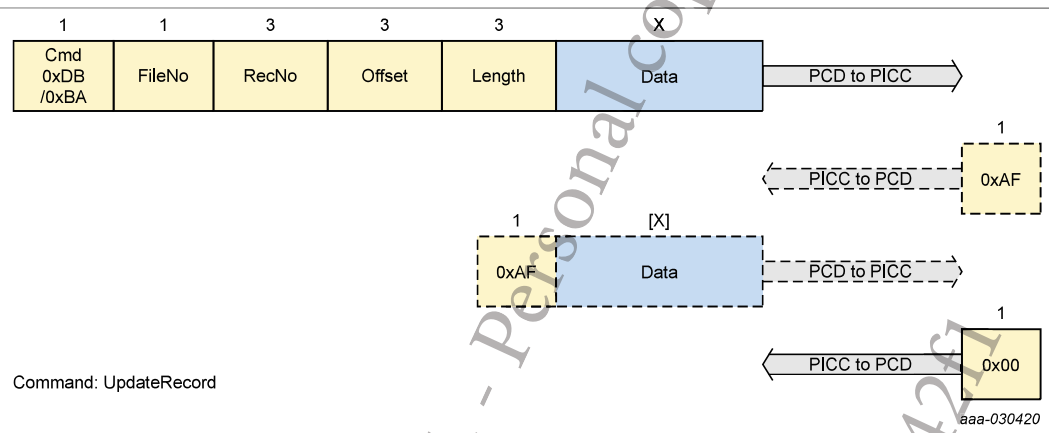


Figure 75. UpdateRecord command protocol

Table 278. Command description - UpdateRecord

UpdateRecord	
Description:	Updates data of an existing record in a FileType.LinearRecord or FileType.CyclicRecord file.
CommMode:	CommMode of targeted file

Table 279. Command parameters description - UpdateRecord

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xDB/0xBA	Command code, 0xDB supports native chaining, while 0xBA supports larger FSC and ISO/IEC 14443-4 chaining
FileNo	1	-	File number of the targeted file
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Full Range	
RecNo	3	-	Record number. Records are numbered starting with the latest record written
		0x000000 .. (ExistingRecCount - 1)	0x000000 meaning the latest record written. The number is limited by the number of existing records (ExistingRecCount)
Offset	3	0x000000 .. (RecordSize - 1)	Starting position for the update operationwithin the targeted record

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 279. Command parameters description - UpdateRecord...*continued*

Name	Length	Value	Description
Length	3	0x000001 .. (RecordSize - Offset)	Number of bytes to be updated.
Command Data Parameters			
Data	X	Full Range	Data to be written in the targeted record

Table 280. Return code description - UpdateRecord

Status	Value	Description
ADDITIONAL_FRAME	0xAF	Only for 0xDB variant: If the number of bytes to send does not fit into one single frame, the PCD has to wait for a status frame with status byte 0xAF before sending the next frame to the PICC
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC or encryption padding
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		Targeted file is not of FileType.LinearRecord or FileType.CyclicRecord
		FileAR.ReadWrite of targeted file only have access conditions set to 0xF
		A ClearRecordFile targeting the currently targeted file has already been performed within the current transaction
		A WriteRecord targeting the currently targeted file has already been performed within the current transaction
		An UpdateRecord targeting a record different from the currently targeted record has already been performed within the current transaction
		SAI given but no 2nd application selected
FILE_NOT_FOUND	0xF0	Targeted file does not exist in the targeted application
AUTHENTICATION_ERROR	0xAE	FileAR.ReadWrite of targeted file not granted while at least one of the access conditions is different from 0xF
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting the required access rights

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-0a2c-4b8c-b0fe-6ecc3cae4251

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 280. Return code description - UpdateRecord...*continued*

Status	Value	Description
BOUNDARY_ERROR	0xBE	RecNo is strictly bigger than the number of existing record in the file
		Offset + 1 is strictly bigger than the record size of the file
		Offset + Length + 1 is strictly bigger than the record size of the file
		Targeted file is of FileType.LinearRecord which is full, i.e. no new records can be written anymore

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.8.10 ClearRecordFile

The detailed description of this command can be found in Section 7.13.3.4.

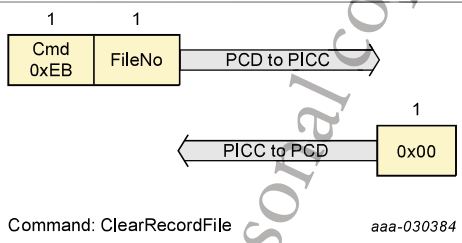


Figure 76. ClearRecordFile command protocol

Table 281. Command description - ClearRecordFile

ClearRecordFile	
Description:	Clear all records in a FileType.LinearRecord or FileType.CyclicRecord file.
CommMode:	CommMode.MAC

Table 282. Command parameters description - ClearRecordFile

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xEB	Command code
FileNo	1	-	File number of the targeted file
	Bit 7		Second Application Indicator
		0b	Target primary application
		1b	Target secondary application
	Bit 6-5		RFU
		00b	
	Bit 4-0		File number
		Full Range	
Command Data Parameters			
-	-	-	No data parameters

Table 283. Return code description - ClearRecordFile

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed

Provided under NDA only
COMPANY CONFIDENTIAL
Sergi Salata
Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-22c-405c-90e-ecc3cae42f1
Personal copy for:

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 283. Return code description - ClearRecordFile*...continued*

Status	Value	Description
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		Targeted file is not of FileType.LinearRecord or FileType.CyclicRecord
		FileAR.ReadWrite of targeted file only have access conditions set to 0xF
		SAI given but no 2nd application selected
FILE_NOT_FOUND	0xF0	Targeted file does not exist in the targeted application
AUTHENTICATION_ERROR	0xAE	FileAR.ReadWrite of targeted file not granted while at least one of the access conditions is different from 0xF
CERT_ERROR	0xCE	Active ECC-based authentication, but CertAccessRights are not granting the required access rights

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f0

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.9 Transaction Management

8.9.1 CommitTransaction

The detailed description of this command can be found in Section 7.14.2.1.

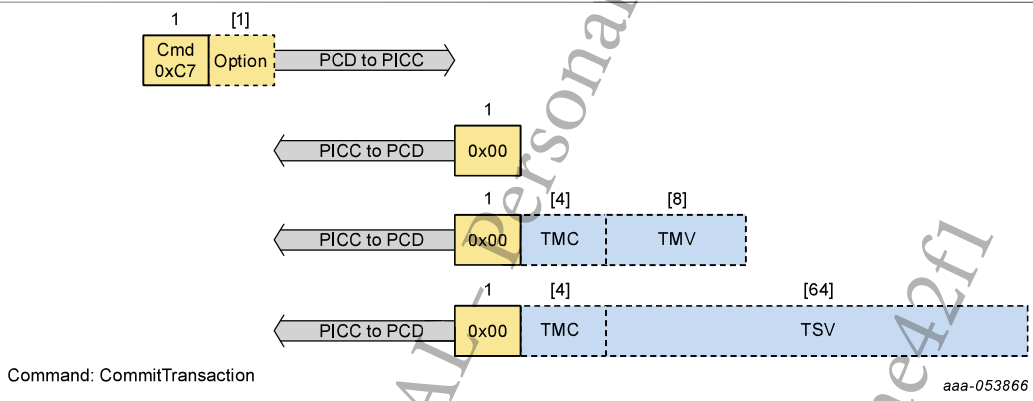


Figure 77. CommitTransaction command protocol

Table 284. Command description - CommitTransaction

CommitTransaction	
Description:	Validates all previous write accesses on FileType.BackupData , FileType.Value , FileType.LinearRecord and FileType.CyclicRecord files within the selected application(s). If applicable, the FileType.TransactionMAC file is updated with the calculated Transaction MAC or Transaction Signature.
CommMode:	CommMode.MAC

Table 285. Command parameters description - CommitTransaction

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xC7	Command code
Option	[1]	-	[Optional] Option parameter
	Bit 7-1		RFU
		0000000b	
	Bit 0		Calculated Transaction MAC/Signature requested on response
		0b	No TMC and TMV returned
		1b	TMC and TMV / TSV returned
Command Data Parameters			
-	-	-	No data parameters

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-cc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 286. Return code description - CommitTransaction

Status	Value	Description
OPERATION_OK	0x00	
		[if Transaction MAC enabled]
		[if Transaction Signature enabled]
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
		TMC and TMV / TSV are requested while not configured for Transaction MAC or Signature calculation is ongoing
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		CommitReaderID is required but was not performed, i.e. TMRICur is the empty string
		TMCLimit was reached.
		Application holds FileType.TransactionMAC configured for Transaction Signature, but KeyID.AppTransactionSIGKey does not exist or is not enabled for Transaction Signature
		Application holds FileType.TransactionMAC configured for Transaction Signature, but KeyID.AppTransactionSIGKey has enabled KeyUsageCtrLimit reached
NO_CHANGES	0x0C	No backup files (FileType.BackupData , FileType.Value , FileType.LinearRecord or FileType.CyclicRecord) updated and (if Transaction MAC/Signature active) TMI is the empty string

Table 287. Response data parameters description - CommitTransaction - OPERATION_OK

Name	Length	Value	Description
TMC	[4]	0x000000 01 .. 0xFF FFFFFF	[Optional, only present if Bit 0 of Option is set] Transaction MAC Counter (TMC)
TMV	[8]	Full Range	[Optional, only present if Bit 0 of Option is set] Transaction MAC Value (TMV)

Table 288. Response data parameters description - CommitTransaction - OPERATION_OK

Name	Length	Value	Description
TMC	[4]	0x000000 01 .. 0xFF FFFFFF	[Optional, only present if Bit 0 of Option is set] Transaction MAC Counter (TMC)
TSV	[64]	Full Range	[Optional, only present if Bit 0 of Option is set] Transaction Signature Value (TSV)

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Salata
Quantas IT Solutions GmbH
Do not copy, share or reproduce
5fc07791-22c440-5c90fe4ec3ae4ff1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.9.2 AbortTransaction

The detailed description of this command can be found in Section 7.14.2.2.

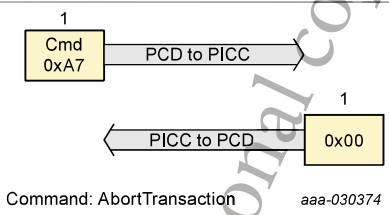


Figure 78. AbortTransaction command protocol

Table 289. Command description - AbortTransaction

AbortTransaction	
Description:	Aborts all previous write accesses on FileType.BackupData , FileType.Value , FileType.LinearRecord and FileType.CyclicRecord files within the selected application(s). If applicable, the Transaction MAC calculation is aborted.
CommMode:	CommMode.MAC

Table 290. Command parameters description - AbortTransaction

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xA7	Command code
Command Data Parameters			
-	-	-	No data parameters

Table 291. Return code description - AbortTransaction

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC (only EV2)
LENGTH_ERROR	0x7E	Command size not allowed
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
NO_CHANGES	0x0C	No backup files (FileType.BackupData , FileType.Value , FileType.LinearRecord or FileType.CyclicRecord) updated and (if Transaction MAC active) TMI is the empty string

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-222c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.9.3 CommitReaderID

The detailed description of this command can be found in Section 7.14.3.4.4.

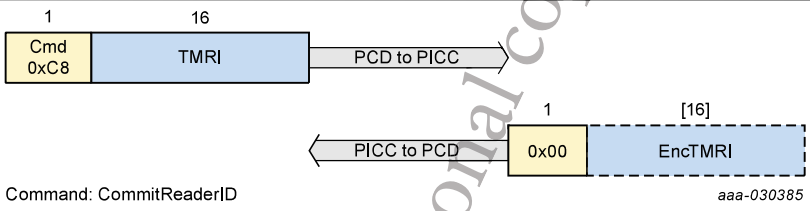


Figure 79. CommitReaderID command protocol

Table 292. Command description - CommitReaderID

CommitReaderID	
Description:	Commits a ReaderID for the ongoing transaction. This will allow a backend to identify the attacking merchant in case of fraud detected.
CommMode:	CommMode.MAC

Table 293. Command parameters description - CommitReaderID

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xC8	Command code
Command Data Parameters			
TMRI	16	Full Range	Transaction MAC ReaderID

Table 294. Return code description - CommitReaderID

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed
PERMISSION_DENIED	0x9D	AID 0x000000 is selected
		None of the selected application(s) hold(s) a FileType.TransactionMAC file
		No ReaderID required according to FileType.TransactionMAC file's access rights
		ReaderID has already been comitted during the ongoing transaction, i.e. TMRICur is not the empty string anymore
AUTHENTICATION_ERROR	0xAE	No active authentication with KeyID.AppCommitReaderIDKey
CERT_ERROR	0xCE	Active ECC-based authentication not granting KeyID.AppCommitReaderIDKey access rights

Provided under NDA only
COMPANY CONFIDENTIAL. Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 295. Response data parameters description - CommitReaderID - OPERATION_OK

Name	Length	Value	Description
EncTMRI	[16]	Full Range	[Optional, present if authenticated] Encrypted Transaction MAC ReaderID of the latest successful transaction

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.10 Cryptographic Support

8.10.1 CryptoRequest

The detailed description of this command can be found in Section 7.15.

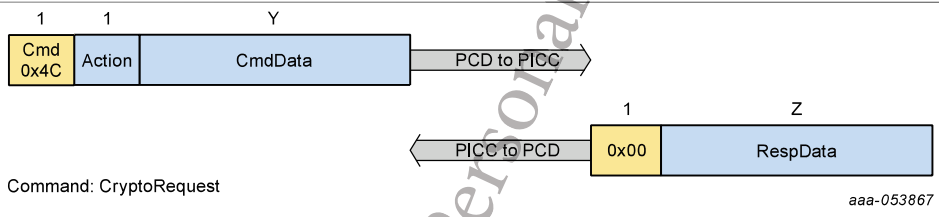


Figure 80. CryptoRequest command protocol

Table 296. Command description - CryptoRequest

CryptoRequest	
Description:	Executes a cryptographic operation. This is the generic API definition, including common error codes. Specific operations are further defined by dedicated subcommands.
CommMode:	CommMode of CryptoRequest as defined by SetConfiguration 0x15.

Table 297. Command parameters description - CryptoRequest

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x4C	Command code
Command Data Parameters			
Action	1	-	Targeted action
		0x03	ECC Sign, see CryptoRequest_ECCSign
CmdData	Y	-	Action specific command data

Table 298. Return code description - CryptoRequest

Status	Value	Description
OPERATION_OK	0x00	
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Integrity error in cryptogram or invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed.
		Invalid action
PERMISSION_DENIED	0x9D	CryptoRequest access condition set to 0xF
AUTHENTICATION_ERROR	0xAE	CryptoRequest access condition not granted while different from 0xF

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-022c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 298. Return code description - CryptoRequest*...continued*

Status	Value	Description
CERT_ERROR	0xCE	Active ECC-based authentication while CryptoRequest access condition not granted while different from 0xF

Table 299. Response data parameters description - CryptoRequest - OPERATION_OK

Name	Length	Value	Description
RespData	Z	-	Action specific response data

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.10.2 CryptoRequest_ECCSign

The detailed description of this command can be found in Section 7.15.1.

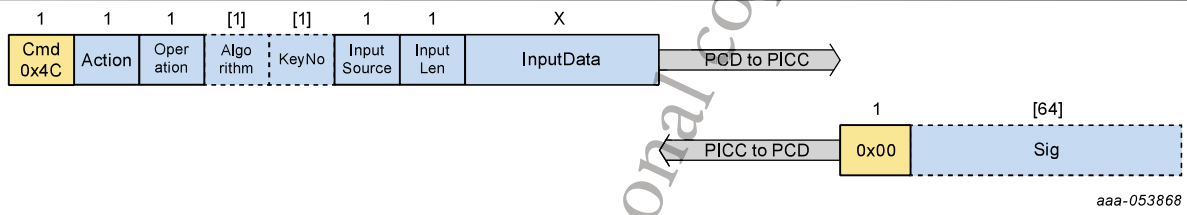


Figure 81. CryptoRequest_ECCSign command protocol

Table 300. Command description - CryptoRequest_ECCSign

CryptoRequest_ECCSign	
Description:	Executes an ECC signature generation. This is a sub-operation of CryptoRequest_ECCSign . For common error conditions, see there.

Table 301. Command parameters description - CryptoRequest_ECCSign

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x4C	Command code
Command Data Parameters			
Action	1	-	Targeted action
		0x03	ECC Sign, see CryptoRequest_ECCSign
Operation	1	-	Targeted operation
		0x01	Initialize signature operation
		0x02	Update data to be signed
		0x03	Finalize signature operation
		0x04	One-shot operation with raw data
		0x05	One-shot operation with pre-computed hash
Algorithm	[1]	-	[Optional, present if Operation = 0x01, 0x04 or 0x05] Targeted algorithm
		0x00	ECDSA with SHA256
KeyNo	[1]	-	[Optional, present if Operation = 0x01, 0x04 or 0x05] Key number of the targeted key
		0x0..0x1	[if AID 0x000000 selected] At PICC level, two keys are supported
		0x0..0x4	[if AID 0x000000 is not selected] At application level, up to five keys are supported
InputSource	1	-	Input Source
		0xF0	Comand Buffer with explicit length
InputLen	1	-	Input Length

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 301. Command parameters description - CryptoRequest_ECCSign...*continued*

Name	Length	Value	Description
		0x00 .. 0xEA	[Operation != 0x05] Raw InputData length
		0x20	[Operation = 0x05] SHA256 length
InputData	X	-	Input Data
		Full range	Hash or raw data bytes

Table 302. Return code description - CryptoRequest_ECCSign

Status	Value	Description
OPERATION_OK	0x00	
LENGTH_ERROR	0x7E	Operation is 0x05 and the InputLen is not 0x20 bytes
NO_SUCH_KEY	0x40	Targeted key does not exist
PERMISSION_DENIED	0x9D	ECDSA Sign disabled over NFC interface
		ECDSA Sign disabled over I^2C interface
		ECDSA Sign disabled via the key policy of the targeted key
		KeyUsageCtrLimit enabled for targeted key has been reached
		Operation is 0x02 or 0x03 while it was not initialized

Table 303. Response data parameters description - CryptoRequest_ECCSign - OPERATION_OK

Name	Length	Value	Description
Sig	[64]	Full range	[Optional, present if Operation = 0x01, 0x04 or 0x05] ECDSA Signature

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.10.3 CryptoRequest_Echo

The detailed description of this command can be found in Section 7.15.2.

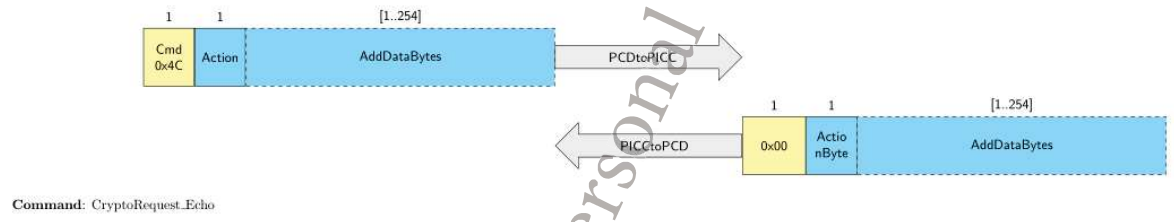


Figure 82. CryptoRequest_Echo command protocol

Table 304. Command description - CryptoRequest_Echo

CryptoRequest_Echo	
Description:	Echos provided command data

Table 305. Command parameters description - CryptoRequest_Echo

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x4C	Command code
Command Data Parameters			
Action	1	-	Targeted action
		0xFD	Echo
AddDataBytes	[1..254]	-	Input Data
		Full range	Additional bytes to echo.

Table 306. Return code description - CryptoRequest_Echo

Status	Value	Description
OPERATION_OK	0x00	

Table 307. Response data parameters description - CryptoRequest_Echo - OPERATION_OK

Name	Length	Value	Description
ActionByte	1	0xFD	Echo operation byte
AddDataBytes	[1..254]	Full range	Additional bytes received in the command data

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantum IT Solutions GmbH
Do not copy, shared or reproduce
5fc07790-22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.11 GPIO Management

8.11.1 ManageGPIO

The detailed description of this command can be found in Section 7.16.2.

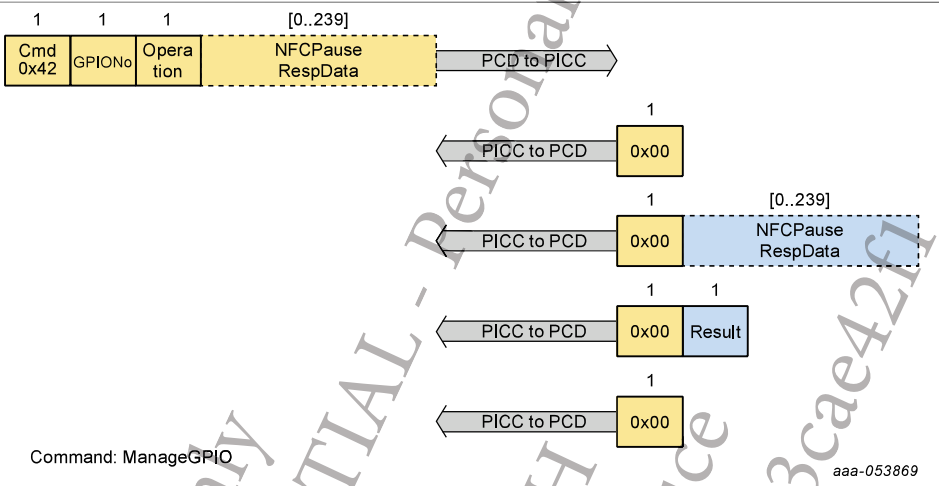


Figure 83. ManageGPIO command protocol

Table 308. Command description - ManageGPIO

ManageGPIO	
Description:	Manages the GPIO output
CommMode:	CommMode of ManageGPIO as defined by SetConfiguration 0x11

Table 309. Command parameters description - ManageGPIO

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x42	Command code
GPIONo	1	-	GPIO Number
		0x00	GPIO1
		0x01	GPIO2
Operation	1	-	Targeted operation
	Bit 7		[if GPIOxMode is output] NFC Control
		0b	No NFC action
		1b	[if over NFC] Pause NFC
		1b	[if over I^2C] Release NFC Pause
	Bit 6-2		[if GPIOxMode is output] RFU
		00000b	
	Bit 1-0		[if GPIOxMode is output] GPIO Control

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 309. Command parameters description - ManageGPIO*...continued*

Name	Length	Value	Description
		00b	CLEAR: clear the GPIO state to LOW (not driven)
		01b	SET: set the GPIO State to HIGH (driven)
		10b	TOGGLE: toggle the GPIO state
		11b	RFU
	Bit 7		[if GPIOxMode is down-stream power out AND Bit7 = '0'] Targeted operation
		0b	Power downstream.
	Bit 6-2		[if GPIOxMode is down-stream power out AND Bit7 = '0'] Target voltage/current level
		0x00	Default level as configured with SetConfiguration
		0x01	Power downstream voltage of 1.8 V and current of 100 µA
		0x02	Power downstream voltage of 1.8 V and current of 300 µA
		0x03	Power downstream voltage of 1.8 V and current of 500 µA
		0x04	Power downstream voltage of 1.8 V and current of 1 mA
		0x05	Power downstream voltage of 1.8 V and current of 2 mA
		0x06	Power downstream voltage of 1.8 V and current of 3 mA
		0x07	Power downstream voltage of 1.8 V and current of 5 mA
		0x08	Power downstream voltage of 1.8 V and current of 7 mA
		0x09	Power downstream voltage of 1.8V and current of 10 mA
		0x0F	Power downstream voltage of 1.8 V and maximal available current
		0x11	Power downstream voltage of 2 V and current of 100 µA
		0x12	Power downstream voltage of 2 V and current of 300 µA
		0x13	Power downstream voltage of 2 V and current of 500 µA
		0x14	Power downstream voltage of 2 V and current of 1 mA
		0x15	Power downstream voltage of 2 V and current of 2 mA
		0x16	Power downstream voltage of 2 V and current of 3 mA
		0x17	Power downstream voltage of 2 V and current of 5 mA
		0x18	Power downstream voltage of 2 V and current of 7 mA
		0x19	Power downstream voltage of 2 V and current of 10 mA
		0x1F	Power downstream voltage of 2 V and maximal available current
	Bit 1		[if GPIOxMode is down-stream power out AND Bit 7 = '0'] GPIO Measurement Control
		0b	No measurement
		1b	MEASURE: execute measurement
	Bit 0		[if GPIOxMode is down-stream power out] GPIO Control
		0b	CLEAR: stop power downstream

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-222c-405c-a0fe-6ec3ca2211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 309. Command parameters description - ManageGPIO*...continued*

Name	Length	Value	Description
		1b	SET: enable power downstream
	Bit 7		[if GPIOxMode is downstream power out AND Bit 7 = '1'] Targeted operation
		1b	Auto WTX
	Bit 6-1		[if GPIOxMode is down-stream power out AND Bit 7 = '1'] RFU
		000000b	
	Bit 0		[if GPIOxMode is down-stream power out AND Bit 7 = '1'] Targeted operation
		0b	CLEAR: stop Auto WTX
		1b	SET: enable Auto WTX
NFCPauseRespData	[0 .. 239]	Full range	[Optional, present if GPIOXMode is output AND Operation[b7] == '1' AND issued over I^2C] NFC Pause Response Data: data to be returned to NFC host in the case of `Release NFC Pause'
Command Data Parameters			
-	-	-	No data parameters

Table 310. Return code description - ManageGPIO

Status	Value	Description
OPERATION_OK	0x00	[if GPIOXMode is output AND (Operation[b7] == '0' OR (Operation[b7] == '1' AND issued over I^2C))]
		[if GPIOXMode is output AND Operation[b7] == '1' AND issued over NFC]
		[if GPIOXMode is down-stream power out AND MEASURE requested]
		[else]
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Integrity error in cryptogram or invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed
PARAMETER_ERROR	0x9E	Parameter value not allowed
PERMISSION_DENIED	0x9D	At PICC level, ManageGPIOAccessCondition is configured for no access (0x0F) and not targeting NFC Pause Release
		At application level, addAID is not set to the currently selected application
		At application level, addManageGPIOAccessCondition is configured for no access (0x0F) and not targeting NFC Pause Release
		Release NFC Pause operation and the NFC interface isn't paused

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-222c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 310. Return code description - ManageGPIO*...continued*

Status	Value	Description
		Targeting GPIO1 while it is not configured for output or down-stream power out
		Enabling down-stream power out while not powered over NFC
		Targeting GPIO2 while it is not configured for output or output with NFCPause file
		Triggering execution of MEASURE while down-stream power out was already enabled
		Enabling Auto WTX while power downstream is not enabled and not powered over V_{CC}
AUTHENTICATION_ERROR	0xAE	At PICC level, no active authentication granting the ManageGPIOAccessCondition while different from 0x0F and not targeting NFC Pause Release
		At application level, no active authentication granting the addManageGPIOAccessCondition while different from 0x0F and not targeting NFC Pause Release
CERT_ERROR	0xCE	At PICC level, active ECC-based authentication while ManageGPIOAccessCondition not granted while different from 0xF and not targeting NFC Pause Release
		At application level, active ECC-based authentication while addManageGPIOAccessCondition not granted while different from 0xF and not targeting NFC Pause Release
WEAK_FIELD	0x1F	Enabling down-stream power out while NFC field strength does not allow the targeted voltage/current selection
PAD_VOLTAGE_UNRELIABLE	0x2F	When enabling power downstream, pad voltage is too low. Pads are not sufficiently supplied

Table 311. Response data parameters description - ManageGPIO - OPERATION_OK

Name	Length	Value	Description
NFCPauseRespData	[0 .. 239]	Full range	NFC Pause Response Data: data received from the I^2C interface

Table 312. Response data parameters description - ManageGPIO - OPERATION_OK

Name	Length	Value	Description
Result	1	Full range	Measurement result

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantagil Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22-405c-9068-5ccaea421

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.11.2 ReadGPIO

The detailed description of this command can be found in Section 7.16.3.

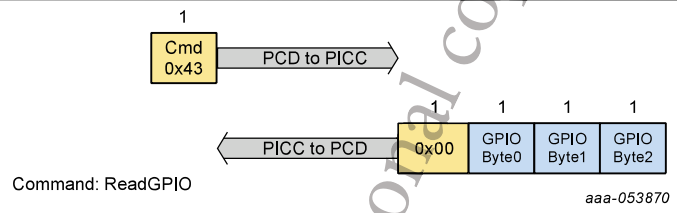


Figure 84. ReadGPIO command protocol

Table 313. Command description - ReadGPIO

ReadGPIO	
Description:	Returns the GPIO statuses
CommMode:	CommMode of ReadGPIO as defined by SetConfiguration 0x11.

Table 314. Command parameters description - ReadGPIO

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0x43	Command code
Command Data Parameters			
-	-	-	No data parameters

Table 315. Return code description - ReadGPIO

Status	Value	Description
OPERATION_OK	0x00	GPIOStatus bytes as defined in Table 47
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
INTEGRITY_ERROR	0x1E	Integrity error in cryptogram or invalid secure messaging MAC
LENGTH_ERROR	0x7E	Command size not allowed
PERMISSION_DENIED	0x9D	At PICC level, ReadGPIOAccessCondition is configured for no access (0x0F)
		At application level, addAID is not set to the currently selected application
		At application level, addReadGPIOAccessCondition is configured for no access (0x0F)
AUTHENTICATION_ERROR	0xAE	At PICC level, no active authentication granting the ReadGPIOAccessCondition while different from 0x0F
		At application level, no active authentication granting the addReadGPIOAccessCondition while different from 0x0F

Provided under NDA only - Personal copy for: COMPANY CONFIDENTIAL Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-0AC-4035-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 315. Return code description - ReadGPIO*...continued*

Status	Value	Description
CERT_ERROR	0xCE	At PICC level, active ECC-based authentication while ReadGPIOAccessCondition not granted while different from 0xF
		At application level, active ECC-based authentication while addReadGPIOAccessCondition not granted while different from 0xF

Table 316. Response data parameters description - ReadGPIO - OPERATION_OK - GPIOStatus bytes as defined in Table 47

Name	Length	Value	Description
GPIOByte0	1	0x49	
GPIOByte1	1	-	
		0x48	[if Input or Output] High
		0x4C	[if Input or Output] Low
		0x49	[else] Invalid
GPIOByte2	1	-	
		0x48	[if Input or Output] High
		0x4C	[if Input or Output] Low
		0x49	[else] Invalid

Provided under NDA only - Personal copy for:
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.12 ISO816-4 Support

8.12.1 ISOSelectFile

The detailed description of this command can be found in Section 7.17.3.4.

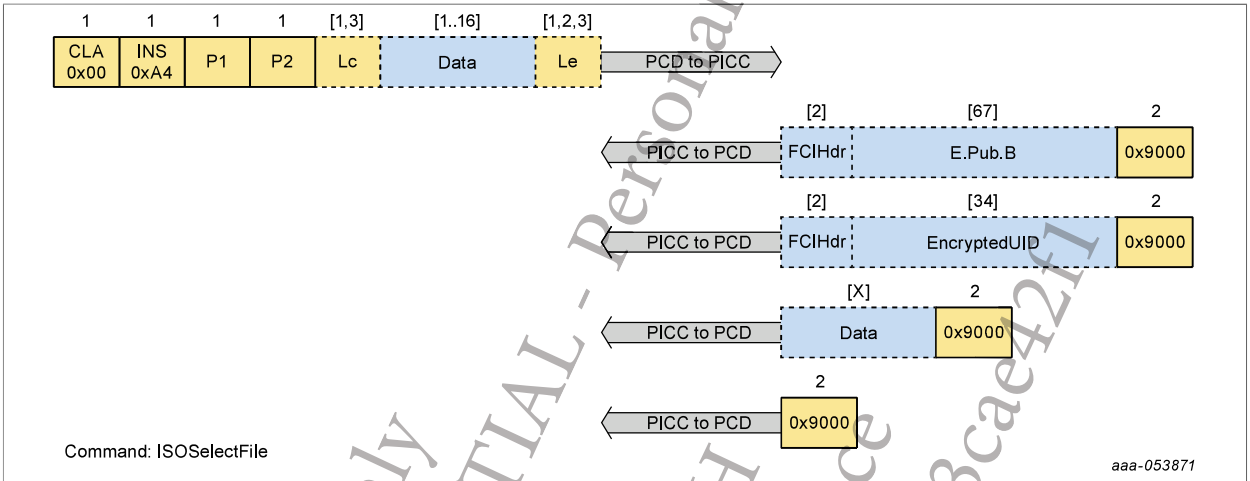


Figure 85. ISOSelectFile command protocol

Table 317. Command description - ISOSelectFile

ISOSelectFile	
Description:	Select an application or file

Table 318. Command parameters description - ISOSelectFile

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0xA4	
P1	1	-	Selection Control
		0x00	Select MF, DF or EF, by file identifier
		0x01	Select child DF
		0x02	Select EF under the current DF, by file identifier
		0x03	Select parent DF of the current DF
		0x04	Select by DF name
P2	1	-	Option
		0x00	Return FCI template
		0x0C	No response data: no FCI shall be returned
Lc	[1, 3]	0x00 .. 0x10	Length of subsequent data field
Data	[1..16]	-	Reference
		Empty	[if P1 == 0x00 OR P1 == 0x03] Select MF

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc079d-c22c405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 318. Command parameters description - ISOSelectFile*...continued*

Name	Length	Value	Description
		Full range	[if P1 == 0x00 OR P1 == 0x01 OR P1== 0x02] Select with the given file identifier
		Full range	[if P1 == 0x04] Select DF with the given DF name
Le	[1, 2, 3]	-	Empty or length of expected response
		Full range	
		0x00	Any expected length (up to 256 byte)
		0x0000	Any expected length (up to 65536 byte)
		0x45 .. 0xFFFF	[if targeted application is configured for integrated ECC-based authentication] Max expected length must be at least 69 bytes.
		0x24 .. 0xFFFF	[else if targeted application is configured for fast UID retrieval] Max expected length must be at least 36 bytes.
		0x01 .. 0xFFFF	[else] Any max expected length allowed.

Table 319. Return code description - ISOSelectFile

Status	Value	Description
ISO9000	0x9000	Correct execution, targeting application that is configured for integrated ECC-based authentication
		Correct execution, targeting application that is configured for fast UID retrieval
		Correct execution, targeting application with FCI in file ID 31
		Correct execution, no FCI requested or configured
ISO6283	0x6283	Application selected with limited functionality: TMCLimit reached
		Application selected with limited functionality: KeyID.AppTransactionSIGKey does not exist or is not enabled for Transaction Signature
		Application selected with limited functionality: KeyID.AppTransactionSIGKey has enabled KeyUsageCtrLimit reached
ISO6700	0x6700	Wrong or inconsistent APDU length
ISO6985	0x6985	Wrapped chained command or multiple pass command ongoing
ISO6A82	0x6A82	Application or file not found, currently selected application remains selected
ISO6A86	0x6A86	Wrong parameter P1 and/or P2
ISO6A87	0x6A87	Wrong parameter Lc inconsistent with P1-P2
ISO6C00	0x6C00	Wrong Le: expected length insufficient for response data

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-0224-4bc0-fe1c-cc3cae2f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 320. Response data parameters description - ISOSelectFile - ISO9000

Name	Length	Value	Description
FCIHdr	[2]	-	[Optional] FCI Header
		T: 0x6F	Tag
		L: 0x43	Length of Value field
E.Pub.B	[67]	-	[Optional] Authentication Data Object: ephemeral public key from PICC
		T: 0x85	Tag
		L: 0x41	Length of Value field
		V: E.Pub.B	Value: ephemeral public key in uncompressed point representation (0x04 || E.Pub.B.x || E.Pub.B.y)

Table 321. Response data parameters description - ISOSelectFile - ISO9000

Name	Length	Value	Description
FCIHdr	[2]	-	[Optional] FCI Header
		T: 0x6F	Tag
		L: 0x22	Length of Value field
EncryptedUID	[34]	-	[Optional] Encrypted UID
		T: 0x85	Tag
		L: 0x20	Length of Value field
		V: RndIV || Payload	Value: - RndIV: 16 byte random - Payload: E(Kx, RndIV XOR VCData)

Table 322. Response data parameters description - ISOSelectFile - ISO9000

Name	Length	Value	Description
Data	[X]	Full Range	[Optional] FCI

Table 323. Response data parameters description - ISOSelectFile - ISO6283 - Application selected with limited functionality: TMCLimit reached

Name	Length	Value	Description
Data	[X]	Full Range	[Optional] FCI options as defined for ISO9000

Table 324. Response data parameters description - ISOSelectFile - ISO6283 - Application selected with limited functionality: KeyID.AppTransactionSIGKey does not exist or is not enabled for Transaction Signature

Name	Length	Value	Description
Data	[X]	Full Range	[Optional] FCI options as defined for ISO9000

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantum Tag IT Solutions GmbH
Do not copy, share or reproduce
5fc07fad-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 325. Response data parameters description - ISOSelectFile - ISO6283 - Application selected with limited functionality: KeyID.AppTransactionSIGKey has enabled KeyUsageCtrLimit reached

Name	Length	Value	Description
Data	[X]	Full Range	[Optional] FCI options as defined for ISO9000

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.12.2 ISOReadBinary

The detailed description of this command can be found in Section 7.17.3.5.

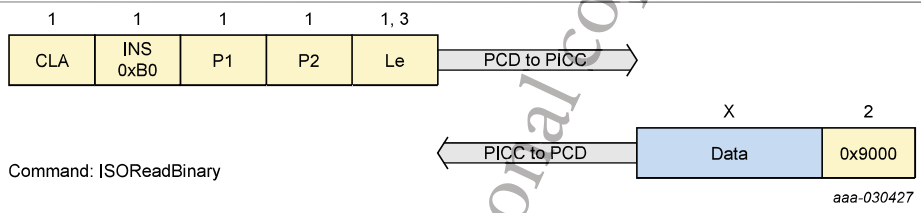


Figure 86. ISOReadBinary command protocol

Table 326. Command description - ISOReadBinary

ISOReadBinary	
Description:	Read from a data file

Table 327. Command parameters description - ISOReadBinary

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0xB0	
P1	1	-	ShortFile ID/Offset
	Bit 7		Encoding
		1b	P1[b 6..5] are RFU P1[b 4..0] encode a short ISO FileID P2[b 7..0] encode an offset from zero to 255
		0b	P1 - P2 (fifteen bits) encode an offset from zero to 32767
	Bit 6-5		[if P1[7] == 1] RFU
		00b	
	Bit 4-0		[if P1[7] == 1] short ISO FileID
		0x00	Targeting currently selected file
		0x01 .. 0x1E	Targeting and selecting file referenced by the given short ISO FileID
		0x1F	RFU
	Bit 6-0		[if P1[7] == 0] Most significant bits of Offset
		(see P2)	
P2	1	0x000000 .. (FileSize - 1)	Offset (see above)
Le	1, 3	-	The number of bytes to be read from the file
		0x000000	Read the entire data file, starting from the position specified in the offset value
		0x000001 .. 0xFFFFFF	If bigger than (FileSize - Offset), the entire data file starting from the offset position is returned
		Full range	

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 328. Return code description - ISOReadBinary

Status	Value	Description
ISO9000	0x9000	Correct execution
ISO6581	0x6581	Memory failure
ISO6700	0x6700	Wrong or inconsistent APDU length
ISO6982	0x6982	Security status not satisfied: no access allowed as FileAR.Read and FileAR.ReadWrite is set to 0xF
		Security status not satisfied: no access allowed as FileAR.Read and FileAR.ReadWrite are different from 0xE and FileAR.SDMFileRead (if SDM enabled) is set to 0xF
		Security status not satisfied: VCState.AuthenticatedAES not allowed
		Security status not satisfied: VCState.AuthenticatedECC not allowed
		Security status not satisfied: SDMReadCtr overflow
		Security status not satisfied: Targeted file cannot be read in VCState.NotAuthenticated as the related SDMReadCtr is equal or bigger than its SDMReadCtrLimit
ISO6985	0x6985	Wrapped chained command or multiple pass command ongoing
		No file selected
		Targeted file is not of FileType.StandardData or FileType.BackupData
		Application of targeted file holds a FileType.TransactionMAC file
		Attempt to read outside file boundaries
		Trying to read SDMSIG while the KeyUsageCtrLimit of the targeted key entry is enabled and reached
		Trying to read SDMSIG while the key configured with FileAR.SDMFileRead2 does not exist or is not enabled for SDM
ISO6A82	0x6A82	File not found
		Wrong application selected after NFCPause release
ISO6A86	0x6A86	Wrong parameter P1 and/or P2

Table 329. Response data parameters description - ISOReadBinary - ISO9000

Name	Length	Value	Description
Data	X	Full range	Data read

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergio Salata
Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fc07791-220-4055-9066-63cc3cce4211

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.12.3 ISOUpdateBinary

The detailed description of this command can be found in Section 7.17.3.6.

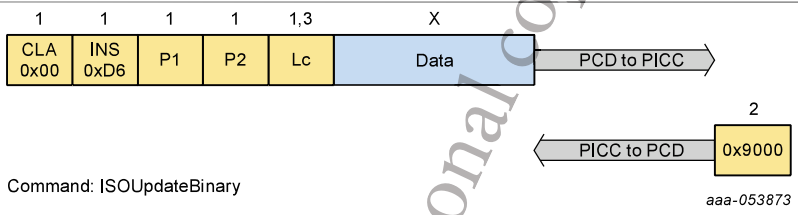


Figure 87. ISOUpdateBinary command protocol

Table 330. Command description - ISOUpdateBinary

ISOUpdateBinary	
Description:	Write to a data file

Table 331. Command parameters description - ISOUpdateBinary

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0xD6	
P1	1	-	ShortFile ID/Offset
	Bit 7		RFU
		1b	P1[b 6..5] are RFU P1[b 4..0] encode a short ISO FileID P2[b 7..0] encode an offset from zero to 255
		0b	P1 - P2 (fifteen bits) encode an offset from zero to 32767
	Bit 6-5		[if P1[7] == 1] RFU
		00b	
	Bit 4-0		[if P1[7] == 1] short ISO FileID
		0x00	Targeting currently selected file
		0x01 .. 0x1E	Targeting and selecting file referenced by the given short ISO FileID
		0x1F	RFU
	Bit 6-0		[if P1[7] == 0] Most significant bits of Offset
		(see P2)	
P2	1	0x000000 .. (FileSize - 1)	Offset (see above)
Lc	1, 3	0x000001 .. (FileSize - Offset)	Length of subsequent data field
Data	X	Full range	Data to be written

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 332. Return code description - ISOUpdateBinary

Status	Value	Description
ISO9000	0x9000	Correct execution
ISO6581	0x6581	Memory failure
ISO6700	0x6700	Wrong or inconsistent APDU length

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 332. Return code description - ISOUpdateBinary*...continued*

Status	Value	Description
ISO6982	0x6982	Security status not satisfied: no access allowed as FileAR.Write and FileAR.ReadWrite is set to 0xF
		Security status not satisfied: authentication required as FileAR.Write and FileAR.ReadWrite different from 0xF is not granted
		Security status not satisfied: VCState.AuthenticatedAES not allowed
		Security status not satisfied: VCState.AuthenticatedECC not allowed
ISO6985	0x6985	Wrapped chained command or multiple pass command ongoing
		No file selected
		Targeted file is not of FileType.StandardData or FileType.BackupData
		Attempt to write beyond the file boundary as set during creation
		Application of targeted file holds a FileType.TransactionMAC file
		Trying to update CRLFile
ISO6A82	0x6A82	File not found
ISO6A86	0x6A86	Wrong parameter P1 and/or P2

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.12.4 ISOReadRecord

The detailed description of this command can be found in Section 7.17.3.7.

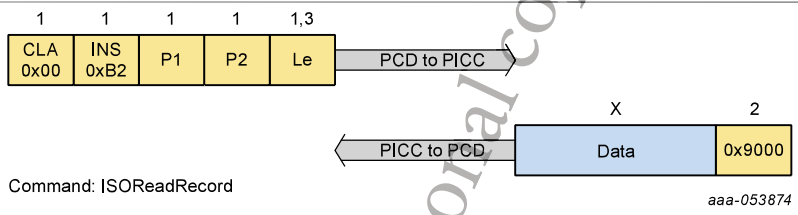


Figure 88. ISOReadRecord command protocol

Table 333. Command description - ISOReadRecord

ISOReadRecord	
Description:	Read from a record file

Table 334. Command parameters description - ISOReadRecord

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0xB2	
P1	1	-	Record Number
		0x00	Current record
		0x01.. ExistingRecCount	Number of first record to be read
		0xFF	RFU
P2	1	-	Reference control
	Bit 7-3		short ISO FileID
		0x00	Targeting currently selected file
		0x01..0x1E	Targeting and selecting file referenced by the given short ISO FileID
		0x1F	RFU
	Bit 2-0		Usage of record number in P1
		100b	Read record P1
		101b	Read all records from P1 up to the last
Le	1, 3	-	The number of bytes to be read
		0x000000	Read the entire file, starting from the record specified
		0x000001 .. 0xFFFFFF	The number of bytes to be read
		Full range	

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-4030-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 335. Return code description - ISOReadRecord

Status	Value	Description
ISO9000	0x9000	Correct execution
ISO6581	0x6581	Memory failure
ISO6700	0x6700	Wrong or inconsistent APDU length
ISO6982	0x6982	Security status not satisfied: no access allowed as FileAR.Read and FileAR.ReadWrite is set to 0xF
		Security status not satisfied: authentication required as FileAR.Read and FileAR.ReadWrite different from 0xF is not granted
		Security status not satisfied: CommMode.Full not allowed
		Security status not satisfied: VCState.AuthenticatedAES not allowed
		Security status not satisfied: VCState.AuthenticatedECC not allowed
ISO6985	0x6985	Wrapped chained command or multiple pass command ongoing
		No file selected
		Targeted file is not of FileType.LinearRecord or FileType.CyclicRecord
		Attempt to read record that has not be written yet or outside file boundaries
		Application of targeted file holds a FileType.TransactionMAC file
ISO6A82	0x6A82	File not found
ISO6A86	0x6A86	Wrong parameter P1 and/or P2
ISO6C00	0x6C00	Le insufficient to sent requested records

Table 336. Response data parameters description - ISOReadRecord - ISO9000

Name	Length	Value	Description
Data	X	Full range	Data read

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.12.5 ISOAppendRecord

The detailed description of this command can be found in Section 7.17.3.8.

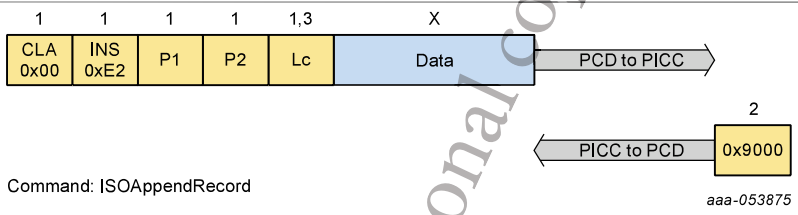


Figure 89. ISOAppendRecord command protocol

Table 337. Command description - ISOAppendRecord

ISOAppendRecord	
Description:	Write a new record to a record file

Table 338. Command parameters description - ISOAppendRecord

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0xE2	
P1	1	-	Record number
		0x00	Current record
P2	1	-	ShortFile ID/Record control
	Bit 7-3		short ISO FileID
		0x00	Targeting currently selected file
		0x01..0x1E	Targeting and selecting file referenced by the given short ISO FileID
		0x1F	RFU
	Bit 2-0		Record control
		000b	First record
Lc	1, 3	Full range	Length of subsequent data field
Data	X	Full range	Record to be written

Table 339. Return code description - ISOAppendRecord

Status	Value	Description
ISO9000	0x9000	Correct execution
ISO6581	0x6581	Memory failure
ISO6700	0x6700	Wrong or inconsistent APDU length

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-22c-40ec-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 339. Return code description - ISOAppendRecord...*continued*

Status	Value	Description
ISO6982	0x6982	Security status not satisfied: no access allowed as FileAR.Write and FileAR.ReadWrite is set to 0xF
		Security status not satisfied: authentication required as FileAR.Write and FileAR.ReadWrite different from 0xF is not granted
		Security status not satisfied: VCState.AuthenticatedAES not allowed
		Security status not satisfied: VCState.AuthenticatedECC not allowed
ISO6985	0x6985	Wrapped chained command or multiple pass command ongoing
		No file selected
		Targeted file is not of FileType.LinearRecord or FileType.CyclicRecord
		Attempt to write beyond the record boundary as set during creation
		UpdateRecord has already been performed within the current transaction
		ClearRecordFile has already been performed within the current transaction
		Targeted file is of FileType.LinearRecord which is full, i.e. no new records can be written anymore
		Application of targeted file holds a FileType.TransactionMAC file
ISO6A82	0x6A82	File not found
ISO6A86	0x6A86	Wrong parameter P1 and/or P2

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-9016-6cc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.12.6 ISOGetChallenge

The detailed description of this command can be found in Section 7.17.3.9.

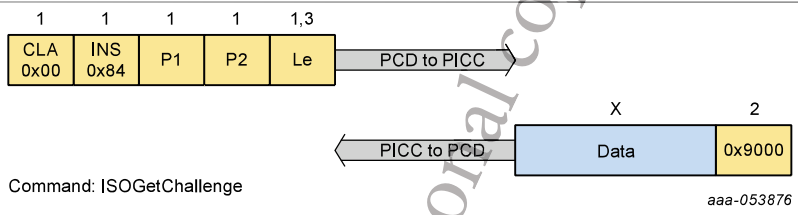


Figure 90. ISOGetChallenge command protocol

Table 340. Command description - ISOGetChallenge

ISOGetChallenge	
Description:	Get a challenge (first part of ISO7816-4 authentication)

Table 341. Command parameters description - ISOGetChallenge

Name	Length	Value	Description
CLA	1	0x00	
INS	1	0x84	
P1	1	0x00	
P2	1	0x00	
Le	1, 3	0x01 .. 0x00FFFF	Length (X) of the expected random

Table 342. Return code description - ISOGetChallenge

Status	Value	Description
ISO9000	0x9000	Correct execution
ISO6700	0x6700	Wrong or inconsistent APDU length
ISO6982	0x6982	Security status not satisfied: VCState.AuthenticatedAES not allowed
ISO6985	0x6985	Wrapped chained command or multiple pass command ongoing
ISO6A86	0x6A86	Wrong parameter P1 and/or P2
ISO6C00	0x6C00	Wrong Le field: zero not allowed

Table 343. Response data parameters description - ISOGetChallenge - ISO9000

Name	Length	Value	Description
Data	X	Full range	Random output

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.13 Proximity Check

8.13.1 PreparePC

The detailed description of this command can be found in Section 7.18.2.1.

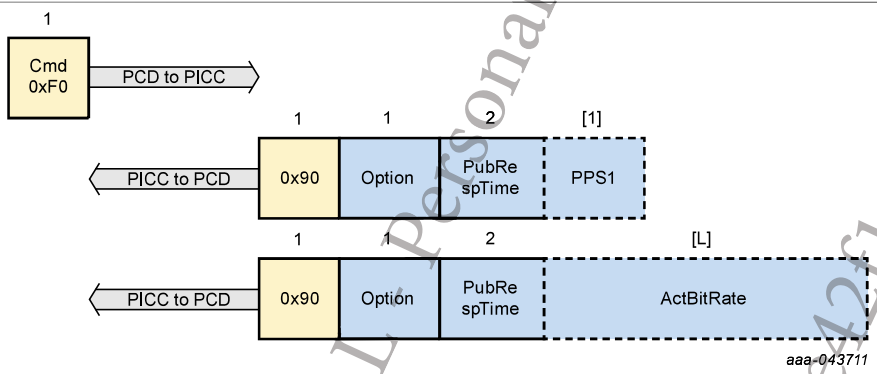


Figure 91. PreparePC command protocol

Table 344. Command description - PreparePC

PreparePC	
Description:	The PD is prepared to perform the Proximity Check by drawing the random number RndR.

Table 345. Command parameters description - PreparePC

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xF0	Command code
Command Data Parameters			
-	-	-	No data parameters

Table 346. Return code description - PreparePC

Status	Value	Description
OK	0x90	
ErrCmdInvalid	0x0B	Command is received when the targeted KeyID.VCProximityKey is still disabled
		Proximity Check is not supported over I^2C
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
ErrFormat	0x0C	Command length different from 1 byte

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 347. Response data parameters description - PreparePC - OK

Name	Length	Value	Description
Option	1	-	Option field defining subsequent response content
	Bit 7-2		RFU
		000000b	
	Bit 1		ActBitRate presence
		0b	Not present
	Bit 0		PPS1 presence
		0b	Not present
		1b	Present
PubRespTime	2	Full range	Published response time (MSB first): time in microseconds. The PD will transmit the ProximityCheck response as close as possible to this time
PPS1	[1]	Limited range	PPS Encoding of the PPS1 byte of [6] . The 4 most significant bits are set to 0 (other values are RFU). The 4 least significant bits indicate the divisor values used to compute the data rate for the PD to PCD and PCD to PD communications. Encoding is the identical as for PPS1 in [6]

Table 348. Response data parameters description - PreparePC - OK

Name	Length	Value	Description
Option	1	-	Option field defining subsequent response content
	Bit 7-2		RFU
		000000b	
	Bit 1		ActBitRate presence
		0b	Not present
		1b	Present
	Bit 0		PPS1 presence
		0b	Not present
PubRespTime	2	Full range	Published response time (MSB first): time in microseconds. The PD will transmit the ProximityCheck response as close as possible to this time
ActBitRate	[L]	Limited range	Active bit rate according to the tag ``A3'' encoding of [6]

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.13.2 ProximityCheck

The detailed description of this command can be found in Section 7.18.2.2.

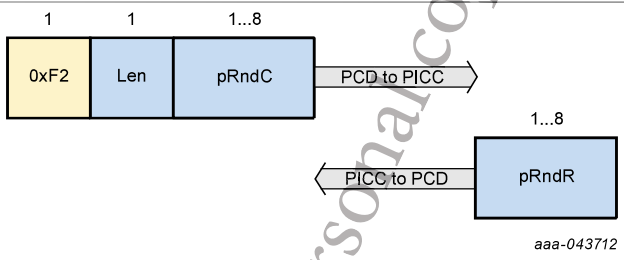


Figure 92. ProximityCheck command protocol

Table 349. Command description - ProximityCheck

ProximityCheck	
Description:	The PD answers with a prepared random number at the published response time in PreparePC . This command may be repeated up to 8 times splitting the random number for different time measurements.

Table 350. Command parameters description - ProximityCheck

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xF2	Command code
Command Data Parameters			
Len	1	0x01..0x08	Length of pRndC
pRndC	1..8	Full range	Part of RndC

Table 351. Return code description - ProximityCheck

Status	Value	Description
OK_PC	N/A	Successful operation. Note that for this command no return code is sent, so only pRndR is returned
ErrCmdInvalid	0x0B	Unexpected command
		Active bit rate changed compared to what was published on PreparePC
		Native command received while preceding PreparePC was ISO/IEC 7816-4 wrapped
		ISO/IEC 7816-4 wrapped command received while preceding PreparePC was native
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
ErrFormat	0x0C	Len is not within valid range
		Actual length of pRndC does not map with Len
ErrCmdOverflow	0x07	Cummulated RndR bytes received exceed 8 bytes

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-022c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 352. Response data parameters description - ProximityCheck - OK_PC

Name	Length	Value	Description
pRndR	1..8	Full range	Part of RndR

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.13.3 VerifyPC

The detailed description of this command can be found in Section 7.18.2.3.

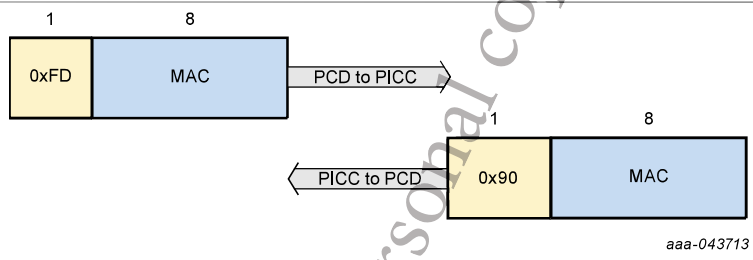


Figure 93. VerifyPC command protocol

Table 353. Command description - VerifyPC

VerifyPC	
Description:	The random numbers are verified using cryptographic methods.

Table 354. Command parameters description - VerifyPC

Name	Length	Value	Description
Command Header Parameters			
Cmd	1	0xFD	Command code
Command Data Parameters			
MAC	8	Full range	MAC computed over the command code, the concatenation of the exchanged random numbers and the information published by the PD in the response of PreparePC

Table 355. Return code description - VerifyPC

Status	Value	Description
OK	0x90	
ErrCmdInvalid	0x0B	Unexpected command
		Active bit rate changed compared to what was published on PreparePC
COMMAND_ABORTED	0xCA	Chained command or multiple pass command ongoing
ErrFormat	0x0C	Command length different from 9 byte

Table 356. Response data parameters description - VerifyPC - OK

Name	Length	Value	Description
MAC	8	Full range	MAC computed over the return code, the concatenation of the exchanged random numbers and the information published by the PD in the response of PreparePC

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.14 EV Charging

8.14.1 VDE_ReadData

The detailed description of this command can be found in Section 7.19.3.1.

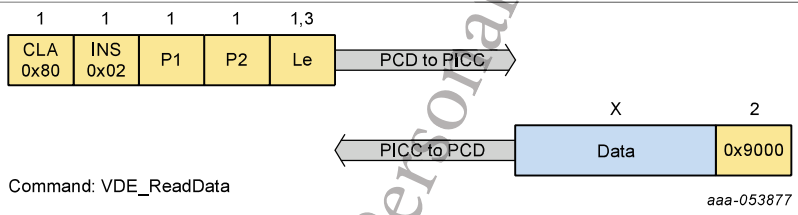


Figure 94. VDE_ReadData command protocol

Table 357. Command description - VDE_ReadData

VDE_ReadData	
Description:	Reads data from FileType.StandardData with FileNo 0x00 or 0x01.

Table 358. Command parameters description - VDE_ReadData

Name	Length	Value	Description
CLA	1	0x80	
INS	1	0x02	
P1	1	-	FileNo of the targeted file
		0x00	
		0x01	
P2	1	0x00	RFU
Le	1,3	-	Length of expected response
		0x00 / 0x000000	Any expected length up to resp. 256/65536 bytes

Table 359. Return code description - VDE_ReadData

Status	Value	Description
ISO9000	0x9000	Correct execution
ISO6700	0x6700	Wrong or inconsistent APDU length
ISO6982	0x6982	Security status not satisfied: VCState.AuthenticatedAES not allowed
		Security status not satisfied: FileAR.Read or FileAR.ReadWrite of targeted FileType.StandardData file only have access conditions set to 0xF
		Security status not satisfied: FileAR.Read or FileAR.ReadWrite of targeted FileType.StandardData file not granted while at least one of the access conditions is different from 0xF

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantum IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-522c-4055-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 359. Return code description - VDE_ReadData*...continued*

Status	Value	Description
ISO6985	0x6985	Conditions of use not satisfied: EV Charging functionality not enabled for the currently selected application
		Conditions of use not satisfied: EV Charging functionality not suppported at PICC level
		Conditions of use not satisfied: targeted application holds a FileType.TransactionMAC
		Conditions of use not satisfied: chained command or multiple pass command ongoing
		Conditions of use not satisfied: targeted file is not of FileType.StandardData
		Conditions of use not satisfied: targeted FileType.StandardData has SDM enable
		Conditions of use not satisfied: targeted FileType.StandardData is a CRLFile
ISO6A82	0x6A82	File or application not found: targeted file does not exist in the targeted application
ISO6A86	0x6A86	Wrong parameter P1: different from 0x00 or 0x01
		Wrong parameter P2: different from 0x00
ISO6C00	0x6C00	Wrong Le: different from 0x00/0x000000

Table 360. Response data parameters description - VDE_ReadData - ISO9000

Name	Length	Value	Description
Data	X	Full range	Data read

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ec3ca4271

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.14.2 VDE_WriteData

The detailed description of this command can be found in Section 7.19.3.2.

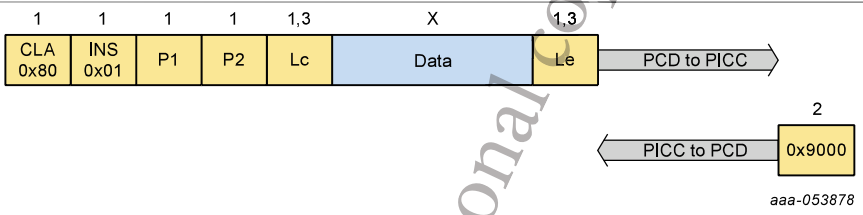


Figure 95. VDE_WriteData command protocol

Table 361. Command description - VDE_WriteData

VDE_WriteData	
Description:	Writes data to FileType.StandardData with FileNo 0x01, and eventually lock the file.

Table 362. Command parameters description - VDE_WriteData

Name	Length	Value	Description
CLA	1	0x80	
INS	1	0x01	
P1	1	0x06	Reserved
P2	1	-	Operation
		0x00	Write
		0x01	Lock
Lc	1, 3	0x000001 .. FileSize	Length of subsequent data field
Data	X	Full range	Data to be written
Le	1,3	-	Length of expected response
		0x00 / 0x000000	Any expected length up to resp. 256/65536 bytes

Table 363. Return code description - VDE_WriteData

Status	Value	Description
ISO9000	0x9000	Correct execution
ISO6700	0x6700	Wrong or inconsistent APDU length
		Data field present while executing Lock operation
ISO6982	0x6982	Security status not satisfied: VCState.AuthenticatedAES not allowed
		Security status not satisfied: write operation (P2=0x00), and FileAR.Write or FileAR.ReadWrite of targeted FileType.StandardData file only have access conditions set to 0xF

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 363. Return code description - VDE_WriteData*...continued*

Status	Value	Description
		Security status not satisfied: lock operation (P2=0x01), and FileAR.Change of targeted FileType.StandardData file only have access conditions set to 0xF
		Security status not satisfied: write operation (P2=0x00), and FileAR.Write or FileAR.ReadWrite of targeted FileType.StandardData file not granted while at least one of the access conditions is different from 0xF
		Security status not satisfied: lock operation (P2=0x01), and FileAR.Change of targeted FileType.StandardData file not granted while different from 0xF
ISO6985	0x6985	Conditions of use not satisfied: EV Charging functionality not enabled for the currently selected application
		Conditions of use not satisfied: EV Charging functionality not suppported at PICC level
		Conditions of use not satisfied: targeted application holds a FileType.TransactionMAC
		Conditions of use not satisfied: chained command or multiple pass command ongoing
		Conditions of use not satisfied: targeted file is not of FileType.StandardData
		Conditions of use not satisfied: targeted FileType.StandardData has SDM enabled
		Conditions of use not satisfied: targeted FileType.StandardData is a CRLFile
		Conditions of use not satisfied: attempt to write beyond the file boundary
ISO6A82	0x6A82	File or application not found: Targeted file does not exist in the targeted application
ISO6A86	0x6A86	Wrong parameter P1: different from 0x06
		Wrong parameter P2: different from 0x00 or 0x01
ISO6C00	0x6C00	Wrong Le: different from 0x00/0x000000

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-40ec-901e-6cc3ce42f

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.14.3 VDE_ECDSASign

The detailed description of this command can be found in Section 7.19.3.3.

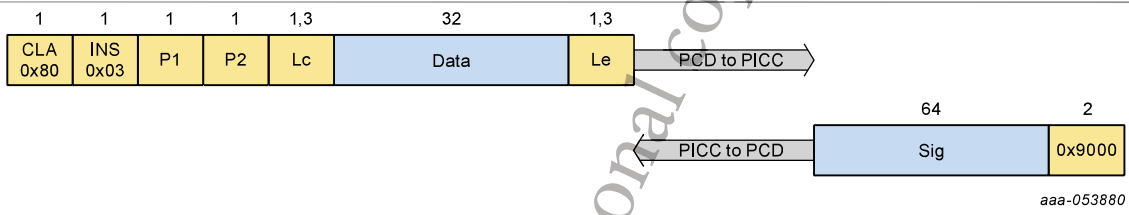


Figure 96. VDE_ECDSASign command protocol

Table 364. Command description - VDE_ECDSASign

VDE_ECDSASign	
Description:	Generates and ECDSA signature over a 32-byte challenge.

Table 365. Command parameters description - VDE_ECDSASign

Name	Length	Value	Description
CLA	1	0x80	
INS	1	0x03	
P1	1	0x0C	Reserved
P2	1	0x09	Reserved
Lc	1, 3	0x20	Length of subsequent data field
Data	32	Full range	32-byte arbitrary data to sign
Le	1,3	-	Length of expected response
		0x00 / 0x000000	Any expected length up to resp. 256/65536 bytes

Table 366. Return code description - VDE_ECDSASign

Status	Value	Description
ISO9000	0x9000	Correct execution
ISO6700	0x6700	Wrong or inconsistent APDU length
ISO6982	0x6982	Security status not satisfied: VCState.AuthenticatedAES not allowed
		Security status not satisfied: CryptoRequest access condition set to 0xF
		Security status not satisfied: CryptoRequest access condition not granted while different from 0xF
ISO6985	0x6985	Conditions of use not satisfied: EV Charging functionality not enabled for the currently selected application
		Conditions of use not satisfied: EV Charging functionality not suppported at PICC level

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 366. Return code description - VDE_ECDSASign...*continued*

Status	Value	Description
		KeyUsageCtrLimit enabled for targeted key has been reached
		Conditions of use not satisfied: chained command or multiple pass command ongoing
		Conditions of use not satisfied: KeyID.ECCPrivateKey with KeyNo 0x00 does not exist
		Conditions of use not satisfied: KeyID.ECCPrivateKey with KeyNo 0x00 has not ECC Sign enabled
		Conditions of use not satisfied: ECDSA Sign disabled over NFC interface
		Conditions of use not satisfied: ECDSA Sign disabled over I^2C interface
ISO6A86	0x6A86	Wrong parameter P1: different from 0x0C
		Wrong parameter P2: different from 0x09
ISO6A87	0x6A87	Unsupported Lc
ISO6C00	0x6C00	Wrong Le: different from 0x00/0x000000

Table 367. Response data parameters description - VDE_ECDSASign - ISO9000

Name	Length	Value	Description
Sig	64	Full range	ECDSA Signature

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cac21f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

8.15 Originality Check

For the asymmetric originality check procedure, the card-unilateral authentication can be used. Refer to ISOInternalAuthenticate.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

9 Limiting values

9.1 Limiting Values applicable to MOA8

Table 368. Limiting values for MOA8[1][2]

In accordance with the Absolute Maximum Rating System (IEC 60134).

Symbol	Parameter	Conditions	Min	Max	Unit
$I_{LA-LB,max}$	maximum input current at LA/LB	-	-	100	mA
V_{ESD}	electrostatic discharge voltage	[3]	-	4	kV
$P_{d,max}$	maximum power dissipation	-	-	80	mW
T_{stg}	storage temperature		-55	125	°C
T_{amb}	ambient temperature for product in MOA8 delivery form		-25	85	°C

[1] Stresses above one or more of the limiting values may cause permanent damage to the device
[2] Exposure to limiting values for extended periods may affect device reliability
[3] ANSI/ESDA/JEDEC JS-001; Human body model: C = 100 pF, R = 1.5 kΩ

9.2 Limiting Values applicable to bumped FFC

Table 369. Limiting values for bumped wafer on FFC[1][2]

In accordance with the Absolute Maximum Rating System (IEC 60134).

Symbol	Parameter	Conditions	Min	Max	Unit
$I_{LA-LB,max}$	maximum input current at LA/LB	-	-	100	mA
V_{ESD}	electrostatic discharge voltage	[3]	-	4	kV
$P_{d,max}$	maximum power dissipation	-	-	80	mW
T_{stg}	storage temperature		-55	125	°C
T_{amb}	ambient temperature for product in bumped wafer on FFC delivery form		-40	105	°C

[1] Stresses above one or more of the limiting values may cause permanent damage to the device
[2] Exposure to limiting values for extended periods may affect device reliability
[3] ANSI/ESDA/JEDEC JS-001; Human body model: C = 100 pF, R = 1.5 kΩ

9.3 Limiting Values applicable to WLCSP and HVQFN20

Table 370. Limiting values for WLCSP and HVQFN20[1][2]

In accordance with the Absolute Maximum Rating System (IEC 60134).

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	supply voltage	[3]	-0.3	2	V
V_I	input voltage	any supply pad [3]	-0.3	2	V
I_{LU}	latch-up current	$V_I < 0$ or $V_I > V_{CC}$	-	-	-
$I_{LA-LB,max}$	maximum input current at LA/LB	-	-	100	mA
V_{ESD}	electrostatic discharge voltage	[4]	-	4	kV

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 370. Limiting values for WLCSP and HVQFN20[1][2] ***...continued***

In accordance with the Absolute Maximum Rating System (IEC 60134).

Symbol	Parameter	Conditions	Min	Max	Unit
$P_{d,max}$	maximum power dissipation	-	-	80	mW
T_{stg}	storage temperature		-55	125	°C
T_{amb}	ambient temperature for product		-40	105	°C

[1] Stresses above one or more of the limiting values may cause permanent damage to the device
[2] Exposure to limiting values for extended periods may affect device reliability
[3] In contact operation between GND and V_{CC} pin
[4] ANSI/ESDA/JEDEC JS-001; Human body model: C = 100 pF, R = 1.5 kΩ

CAUTION

This device is sensitive to ElectroStatic Discharge (ESD). Observe precautions for handling electrostatic sensitive devices.

Such precautions are described in the *ANSI/ESD S20.20, IEC/ST 61340-5, JESD625-A* or equivalent standards.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

10 Recommended operating conditions

MF3E(H)x3 is charactrized by its specified operating field strength range of 1.5 A/m to 7.5 A/m and operating supply voltage range of 1 V to 1.98 V.

10.1 Recommended operating conditions applicable to MOA8

Table 371. Recommended operating conditions for MOA8

Symbol	Parameter	Conditions	Min	Typ[1]	Max	Unit
H	field strength	contactless interface operation	1.5	-	7.5	A/m
T_{amb}	operating ambient temperature for product in MOA8 delivery form	[2]	-25	-	85	°C

[1] Typical values are only referenced for information. They are subject to change without notice
[2] All product properties and values specified within this data sheet are only valid within the specified operating ambient temperature range

10.2 Recommended operating conditions applicable bumped FFC

Table 372. Recommended operating conditions for bumped wafer on FFC

Symbol	Parameter	Conditions	Min	Typ[1]	Max	Unit
H	field strength	contactless interface operation	1.5	-	7.5	A/m
T_{amb}	operating ambient temperature for product in bumped wafer on FFC delivery form	[2]	-40	-	105	°C

[1] Typical values are only referenced for information. They are subject to change without notice
[2] All product properties and values specified within this data sheet are only valid within the specified operating ambient temperature range

10.3 Recommended operating conditions applicable to WLCSP and HVQFN20

Table 373. Recommended operating conditions for WLCSP and HVQFN20

Symbol	Parameter	Conditions	Min	Typ[1]	Max	Unit
V_{CC}	supply voltage	nominal supply voltage [2]	1.0 [3]	-	1.98 [4]	V
V_I	DC input voltage on digital inputs and digital I/O pads	-	-0.3	-	V_{CC} + 0.3 V	V
H	field strength	contactless interface operation	1.5	-	7.5	A/m
T_{amb}	operating ambient temperature for product	[5]	-40	-	105	°C

[1] Typical values are only referenced for information. They are subject to change without notice
[2] The supply voltage operating range of 1 V to 1.98 V requires internal supply elevation for the supply voltage range of 1 V to 1.62 V. The supply voltage mode is automatically selected during boot-up based on internal supply voltage measurement. To avoid continues activation and deactivation of the internal supply voltage elevation the external supply voltage of 1.55 V to 1.62 V should be avoided as performance degradation or resets might occur in this supply voltage range due to internal supply voltage switching. Performance degradation or chip resets might lead to timeouts during I^2C communication. Therefore it is recommended that the host would continue to retry the read for a preset number of times in case of timeouts and after that it will go to recovery mode trying with interface/chip reset and even if there is no response, returns with an error for the application to reopen the session. The V_{CC} supply voltage rise time impacts the power consumption. VCC supply voltage ramp times smaller than 600 µs to 1.8 V lead to higher power consumption as the device boots in voltage elevation mode. For V_{CC} supply voltages >1.62 V the supply voltage ramp shall therefore >600 µs.
[3] 1 V +10 %
[4] 1.8V +/-10 %
[5] All product properties and values specified within this data sheet are only valid within the specified operating ambient temperature range

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc07f0d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

11 Characteristics

11.1 AC Characteristics

AC Characteristics described in this section are applicable to all product delivery types: MOA8, FFC, WLCSP and HVQFN20.

Table 374. Electrical AC Characteristics of antenna pins LA, LB

V_{CC} = 1.00 V to 1.98 V; V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified [1]

Symbol	Parameter	Conditions	Min	Typ[2]	Max	Unit
f_{LALB}[3]	Operating frequency LA, LB	-	-	13.56	-	MHz
C_{LALB} [3]	configured for antenna input with 17 pF capacitance. Test frequency = 13.56 MHz; T_{amb} = 25 °C	$V_{LA,LB}$ = 2.1 V (rms) $V_{LA,LB}$ = 0.3 V (rms)	-	17.6 17.3	-	pF
C_{LALB} [3]	Configured for antenna input with 69 pF capacitance. Test frequency = 13.56 MHz; T_{amb} = 25 °C	$V_{LA,LB}$ = 2.1 V (rms)	-	68.1	-	pF
R_{LALB} [3][4][5][6]	Configured for antenna input with 17 pF capacitance. Test frequency = 13.56 MHz; T_{amb} = 25 °C	$V_{LA,LB}$ = 2.1 V (rms) $V_{LA,LB}$ = 0.3 V (rms)	-	1.4	-	kΩ
R_{LALB} [3][4][5][6]	Configured for antenna input with 69 pF capacitance. Test frequency = 13.56 MHz; T_{amb} = 25 °C	$V_{LA,LB}$ = 2.6 V (rms)	-	1.3	-	kΩ

[1] For the MOA8 delivery form, following values for T_{amb} do apply: T_{amb} = -25 °C to 85 °C
[2] Typical values are only referenced for information. They are subject to change without notice
[3] C_{LALB} and C_{LALB} values stated here assume a parallel RC equivalent circuit for the chip
[4] The value stated here was measured at estimated start of chip operation
[5] Measured with sine wave at L_A, L_B
[6] Parameter is valid in contactless ISO/IEC 14443 compliant operation only

Table 375. Non-Volatile memory timing characteristics

V_{CC} = 1 V to 1.98 V; V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified [1]

Symbol	Parameter	Conditions	Min	Typ[2]	Max	Unit
t_{EEP}	FLASH erase + program time	-	-	-	2.3	ms
t_{EEE}	FLASH erase time	-	-	-	0.9	ms
t_{EEW}	FLASH program time	-	-	-	1.4	ms
t_{EER}	FLASH data retention time	T_{amb} = 55 °C	25	-	-	years
N_{EEC}	FLASH endurance (maximum number of programming cycles applied to the whole memory block performed by NXP static and dynamic wear leveling algorithm)		20 x 10^6	100 x 10^6	-	cycles

[1] For the MOA8 delivery form, following values for T_{amb} do apply: T_{amb} = -25 °C to 85 °C
[2] Typical values are only referenced for information. They are subject to change without notice

Provided under NDA only - Personal copy for:
COMPANY CONFIDENTIAL
Sergii Salata
Quantum Solutions GmbH
Do not copy, share or reproduce
5fc0779d-022c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

11.1.1 AC Characteristics applicable to WLCSP and HVQFN20

Table 376. Electrical AC Characteristics of SDA, SCL

V_{CC} = 1 V to 2 V; V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified

Symbol	Parameter	Conditions	Min	Typ [1]	Max	Unit
tr_{IO} [2][3]	I/O input rise time	input / reception mode	-	-	1	µs
tf_{IO} [4][3]	I/O input fall time	input / reception mode	-	-	1	µs
tf_{OIO}	I/O output rise time	output / transmssion mode; C_L = 30 pF	-	-	0.3	µs
f_{CLK}	External clock fequency in I^2C applications	t_{CLKWL}, T_{amb} and V_{CC} within specified limits	-	-	1.0	MHz
C_{PIN}	Pin capacitances SDA, SCL	Test frequency = 1 MHz; T_{amb} = 25 °C	-	-	10.5	pF
P_{OUT}	Maximum output power in power harvesting mode at GPIO1	-	-	-	10	mW

[1] Typical values are only referenced for information. They are subject to change without notice
[2] tr is defined as rise time between 30 % and 70 % of the signal amplitude
[3] maximum recommend load 50pF
[4] tf is defined as fall time between 70 % and 30 % of the signal amplitude

11.2 DC Characteristics

Measurement conventions: Testing measurements are performed at the contact pads of the device under test. All voltages are defined with respect to the ground contact pad VSS. All currents flowing into the device are considered positive.

11.2.1 DC Characteristics applicable to bumped FFC

There is one general purpose I/O interface available in the bumped FFC delivery type, the GPIO1. Desciptions in this section are applicable to the bumped wafer on FFC delivery form.

11.2.1.1 General purpose I/O interface: GPIO1

Table 377. Electrical DC characteristics of GPIO1

V_{CC} = 1 V to 2 V; V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified

External pullup resistor 20 kΩ to V_{CC} assumed. The worst case test condition for parameter V_{OH} is present at minimum V_{CC}.

Symbol	Parameter	Conditions	Min	Typ[1]	Max	Unit
V_{IH}	HIGH level input voltage	-	0.7 x V_{CC}	-	V_{CC} + 0.3	V
V_{IL}	LOW level input voltage	-	-0.3	-	0.25 x V_{CC}	V
I_{IH}	HIGH level input current in "weak pullup" input mode	0.7 V_{CC} ≤ V_I ≤ V_{CC}; Test conditions for the maximum absolute value: $I_{IH(max)}$: V_I = 0.7 V_{CC}, V_{CC} = $V_{CC(max)}$	-	-1	-20	µs

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c226-405c-90fe-6ecc3cae42f1

Table 377. Electrical DC characteristics of GPIO1...*continued*

V_{CC} = 1 V to 2 V; V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified
External pullup resistor 20 kΩ to V_{CC} assumed. The worst case test condition for parameter V_{OH} is present at minimum V_{CC}.

Symbol	Parameter	Conditions	Min	Typ[1]	Max	Unit
I_{IIL}	LOW level input current	0 V_{CC} ≤ V_I ≤ 0.3 V_{CC}; Test conditions for the maximum absolute value: $I_{IL(max)}$:V_I = 0 V, V_{CC} = $V_{CC(max)}$	-	-1	-50	µs
I_I	input current in "weak pullup" input mode	0 V_{CC} ≤ V_I ≤ V_{CC}; Test conditions for the maximum absolute value: $I_{I(max)}$:V_I = 0 V, V_{CC} = $V_{CC(max)}$	0	-	-50	µs
I_{ILIH}	leakage input current at input voltage beyond V_{CC} in "weak pullup" input mode	V_{CC} < V_I ≤ V_{CC} + 0.3 V; -25 °C ≤ T_{amb} ≤ 85 °C; Test conditions: V_I = V_{CC} + 0.3 V; V_{CC} = $V_{CC(max)}$; T_{amb} = 85 °C	-	-	20	µs
I_{ILIL}	leakage input current at input voltage below V_{SS} in "weak pullup" input mode	V-0.3 V ≤ V_I < 0 V; -25 °C ≤ T_{amb} ≤ 30 °C; Test conditions: V_I = -0.3 V; V_{CC} = $V_{CC(max)}$; T_{amb} = 30 °C	-	-	50	µs
V_{OH}	HIGH level output voltage	I_{OH} = -20 µA	0.7 x V_{CC}	-	-	V
V_{OL}	LOW level output voltage	I_{OL} = 1.0 mA I_{OL} = 0.5 mA	-	-	0.3 0.7 x V_{CC}	V

[1] Typical values are only referenced for information. They are subject to change without notice

11.2.2 DC Characteristics applicable to WLCSP and HVQFN20

11.2.2.1 General purpose I/O interface

Table 378. Eletrical DC characteristics of GPIO1/2

V_{CC} = 1 V to 2 V; V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified
External pullup resistor 20 kΩ to V_{CC} assumed. The worst case test condition for parameter V_{OH} is present at minimum V_{CC}.

Symbol	Parameter	Conditions	Min	Typ[1]	Max	Unit
V_{IH}	HIGH level input voltage	-	0.7 x V_{CC}	-	V_{CC} + 0.3	V
V_{IL}	LOW level input voltage	-	-0.3	-	0.25 x V_{CC}	V

Provided under NDA only. COMPANY CONFIDENTIAL - Personal copy for: Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-405c-90fe-6ecc3cae42f1

Table 378. Eletrical DC characteristics of GPIO1/2...*continued*

V_{CC} = 1 V to 2 V; V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified

External pullup resistor 20 kΩ to V_{CC} assumed. The worst case test condition for parameter V_{OH} is present at minimum V_{CC}.

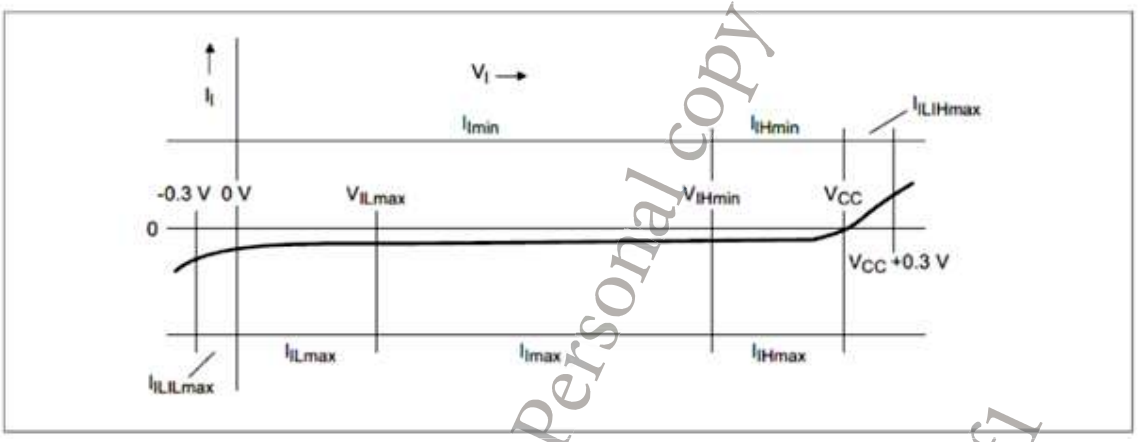
Symbol	Parameter	Conditions	Min	Typ[1]	Max	Unit
I_{IH}	HIGH level input current in "weak pullup" input mode	0.7 V_{CC} ≤ V_I ≤ V_{CC}; Test conditions for the maximum absolute value: $I_{IH(max)}$:V_I = 0.7 V_{CC}, V_{CC} = $V_{CC(max)}$	-	-1	-20	µA
I_{IIL}	LOW level input current	0 V_{CC} ≤ V_I ≤ 0.3 V_{CC}; Test conditions for the maximum absolute value: $I_{IL(max)}$:V_I = 0 V, V_{CC} = $V_{CC(max)}$	-	-1	-50	µA
I_I	input current in "weak pullup" input mode	0 V_{CC} ≤ V_I ≤ V_{CC}; Test conditions for the maximum absolute value: $I_{I(max)}$:V_I = 0 V, V_{CC} = $V_{CC(max)}$	0	-	-50	µA
I_{ILIH}	leakage input current at input voltage beyond V_{CC} in "weak pullup" input mode	V_{CC} < V_I ≤ V_{CC} + 0.3 V; -25 °C ≤ T_{amb} ≤ 85 °C; Test conditions: V_I = V_{CC} + 0.3 V; V_{CC} = $V_{CC(max)}$; T_{amb} = 85 °C	-	-	20	µA
I_{ILIL}	leakage input current at input voltage below V_{SS} in "weak pullup" input mode	V-0.3 V ≤ V_I < 0 V; -25 °C ≤ T_{amb} ≤ 30 °C; Test conditions: V_I = -0.3 V; V_{CC} = $V_{CC(max)}$; T_{amb} = 30 °C	-	-	50	µA
V_{OH}	HIGH level output voltage	I_{OH} = -20 µA	0.7 x V_{CC}	-	0.3 0.7 x V_{CC}	V
V_{OL}	LOW level output voltage	I_{OL} = 1.0 mA I_{OL} = 0.5 mA	-	-	0.3 0.7 x V_{CC}	V

[1] Typical values are only referenced for information. They are subject to change without notice

Provided under NDA only.
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions Gmbh
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-beec3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.



V_{CC} = 1 V to 2 V; V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified

External pullup resistor 20 kΩ to V_{CC} assumed. The worst case test condition for parameter V_{OH} is present at minimum V_{CC}.

Figure 97. Input characteristics of GPIO1/2

11.2.2.2 I²C Interface

Table 379. Electrical DC characteristics of I²C

V_{CC} = 1 V to 2 V; V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified; Pads SCL, SDA are in open-drain mode

Symbol	Parameter	Conditions	Min	Typ[1]	Max	Unit
V_{IH}	HIGH level input voltage	-	0.7 x V_{CC}	-	V_{CC}+ 0.3 V	V
V_{IL}	LOW level input voltage	-	-0.3	-	0.25 x V_{CC}	V
V_{HYS}	input hysteresis voltage	-	0.081	-		V
$V_{OL(OD)}$	low-level output voltage (open-drain mode)	I_{OL} = 3.0 mA V_{CC} >= 1.1 V	0	-	0.4	V
$I_{OL(OD)}$	low-level output current (open-drain mode)	V_{OL} = 0.6 V V_{CC} >= 1.1 V	0.6	-	-	mA
I_{WPU}	weak pullup current	V_{IO} = 0 V	-	-180	-	µA
I_{ILIH}	leakage input curernt high level	V_{SDA} = 3.6 V V_{SCL} = 3.6 V	-	0.27	15	µA

[1] Typical values are only referenced for information. They are subject to change without notice

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

11.2.2.3 Power Consumption

Table 380. Electrical DC characteristics of IC supply voltage V_{CC}

V_{SS} = 0 V; T_{amb} = -40 °C to 105 °C, unless otherwise specified;

Symbol	Parameter	Conditions	Min	Typ [1]	Max	Unit
VCC	supply voltage range [2]	-	1.0	-	2	V
IDD	supply current high-performance mode, CPU in Idle mode and AES or ECC cryptographic operation in parallel	-	-	-	15	mA
IDD	supply current low-performance mode, CPU halted and AES or ECC cryptographic operation	-	-	-	0.65	mA
IDD	supply current HALT mode [3]	-	-	-	5	µA
IDD	Idle [4]	V_{CC} = 1 V - 1.62 V	-	-	1.65	mA
		V_{CC} = 1.62 V - 1.98 V	-	-	6.5	mA

[1] Typical values are only referenced for information. They are subject to change without notice
[2] The supply voltage ramp shall not be faster than 10us for supply voltage ramp-up to 1.98 V.
[3] In order to guarantee the maximum HALT current floating behavior of the GPIO pins shall be avoided. For details refer to Section 7.7.3.1.
[4] For power critical applications, the usage of HALT mode instead of Idle is recommended for power saving reasons.

11.2.2.4 I^2C Bus Timings

The MF3E(H)x3 I^2C bus timing parameters are in accordance to the NXP I^2C bus specification, please refer to [31].

11.2.3 EMC / EMI

EMC and EMI resistance of MF3E(H)x3 is according to ISO/IEC 61967-4, refer to [15].

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

12 Package outline

MF3E(H)x3 is delivered in two main form factors, the sawn 75 µm wafer on FFC and the contactless smartcard module package MOA8.

Sawn wafer on FFC delivery

MF3E(H)x3 is offered in form of a 12-inch wafer delivery (sawn wafer, 75 µm thickness, on film frame carrier; electronic fail die marking according to SECSII format). With this delivery type, four pins are available for the contactless smartcard or contactless tag manufacturing.

A detailed description of the delivery type including pad coordinates is available inside the “Wafer and Delivery Specification”, [36] .

Figure 98. MF3E(H)x3 package outline for bumped sawn wafer on FFC delivery

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergi Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

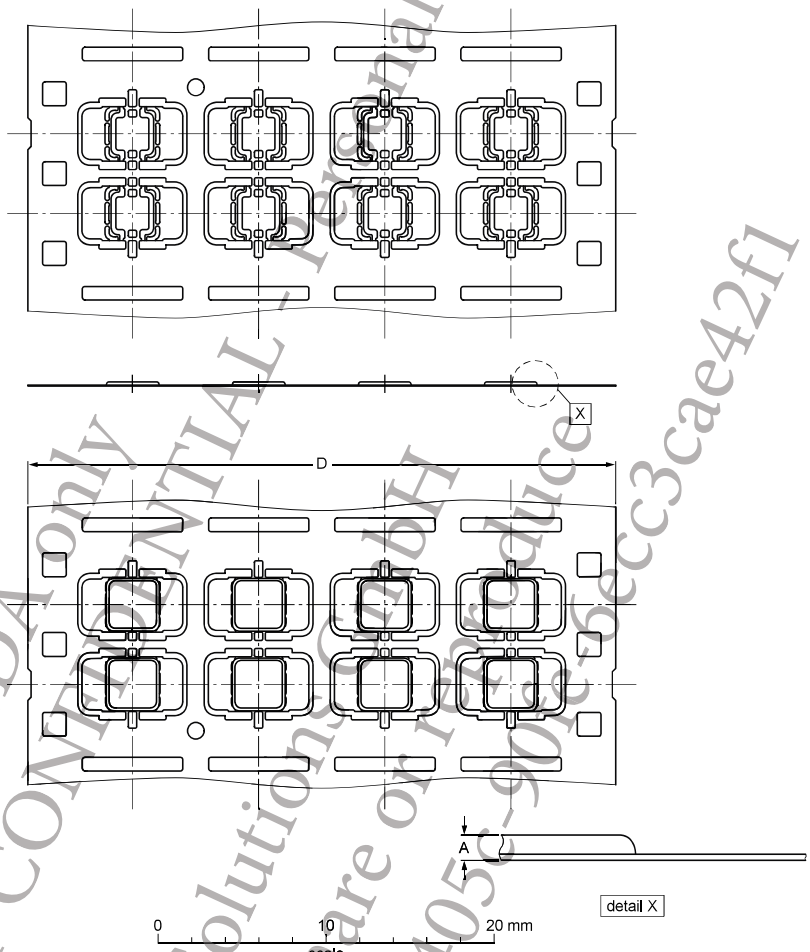
© 2025 NXP B.V. All rights reserved.

MOA8 delivery

MF3E(H)x3 is offered in form of MOA8 contactless smartcard IC modules.

Table 381. MF3E(H)x3 package outline for MOA8 delivery

PLLMC: plastic leadless module carrier package; 35 mm wide tape — SOT500-4



Dimensions

Unit		A(1)	D
mm	max	0.26	35.05
	nom		35.00
	min		34.95

For unspecified dimensions see PLLMC-drawing given in the subpackage code.

Note
1. Total package thickness, exclusive punching burr.

sot500-4_po

Outline version	References				European projection	Issue date
	IEC	JEDEC	JEITA			
SOT500-4	- - -	- - -	- - -			11-02-18

Figure 99. Package outline SOT500-4 (MOA8)

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

13 Abbreviations

Table 382. Abbreviations and symbols

Acronym	Description
2TDEA	Two key triple DES, keying option 2 of [14]
3TDEA	Three key triple DES, keying option 1 of [14]
AES	Advanced Encryption Standard, see [16]
AID	Application IDentification
AKS	Active Key Set
API	Application Programming Interface
APDU	Application Protocol Data Unit
AppKeySettings	The application level key configuration settings
AppKeySetSettings	The application level key set related configuration settings
ATS	Answer To Select
AuthKey	The key of the currently active authentication
Automatic Fare Collection	Paying for fares of, for example, public transport via contactless operations.
BCC	UID CLn check byte, calculated as exclusive-or over the 4 previous bytes, Type A
Card	A Proximity Device (PD) in card form factor, so a PICC.
Card Issuer	The Card Issuer personalizes the card and hands it over to the Card Holder, possibly via some distribution channel.
Card Holder	The Card Holder owns the card after it was personalized.
CBC	Cipher Block Chaining, see [17].
CID	Card IDentifier
CIO	Contactless Infrastructure Operator
CLF	ContactLess Frontend device The chip, antenna and electronic components which take care of the communication between the contactless side and the digital interface. This term is commonly used for the NFC device. In documents, it is also used as the term for the same function in the PCD.
CMAC	Cipher-based MAC, see [18].
CmdCtr	Command Counter
Cmd	Command
CmdState	Command state. The state of the card with regard to command execution.
Contactless interface	Interface between PD and PCD
CommMode	Communication Mode. The communication mode defines the protection applied during secure messaging. MF3E(H)x3 supports three different modes: Plain, MAC and Full.
CRC	Cyclic Redundancy Check
DAM	Delegated Application Management
DES	Data Encryption Standard, see [13].
DF	Dedicated File, see [10].
D(K,X)	Symmetric decryption of ciphertext X using secret key K.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 382. Abbreviations and symbols...*continued*

Acronym	Description
ECC	Elliptic Curve Cryptography
EDC	Error Detection Code, see [6].
EF	Elementary File, see [10].
EncTMRI	Encrypted Transaction MAC ReaderID of the previous transaction.
E(K,X)	Symmetric encryption of plaintext X using secret key K.
EEPROM	Electrically Erasable Programmable Read-Only Memory
Factory Programming	The process performed by the IC Manufacturer or on behalf of the IC Manufacturer to write information into the chip. For example, UID and default configuration options.
FileAR	File Access Right
FileType	Type of a file. The file types supported by MF3E(H)x3 are standard data, backup data, value, linear record, cyclic record, or transaction MAC file.
FSC	Frame Size for Proximity Card, see [6].
FSD	Frame Size for Proximity Coupling Card, see [6].
FDT	Frame Delay Time, see [6].
FFC	Film Frame Carrier
FWT	Frame Waiting Time, see [6].
GPIO	General Purpose Input/Output
IC	Integrated Circuit
IC Manufacturer	The IC Manufacturer takes care about production of the silicon and testing the device together with initializing it with manufacturer data.
IID	Installation Identifier.
INF	Information Field, see [6].
Installation Identifier	A 16 byte number that identifies the Installation of the PCD. It is a number which is an operator must uniquely generate for each installation following a scheme prescribed by NXP. It is used to match with the Virtual Card Installation Identifier to determine whether a certain Virtual Card belongs to a certain installation.
IV	Initialization Vector
ISOL 3	See ISO Layer 3
ISO Layer 3	It refers to the ISO/IEC 14443 Layer 3 [4].
ISOL 4	See ISO Layer 4.
ISO Layer 4	It refers to the ISO/IEC 14443 Layer 4 [6].
KDF	Key Derivation Function
KeyID	Key Identifier. The name used for a key throughout the document.
KeyType	Type of a key. Either 2TDEA, 3TDEA or AES.
KeyVer	Key Version
LCR	L = inductance, Capacitance, Resistance (LCR meter)
LSB	Least Significant Byte
MAC	Message Authentication Code

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 382. Abbreviations and symbols...*continued*

Acronym	Description
MACt	Truncated message authentication code
MAD	MIFARE Application Directory
MF	Master File, see [10].
MFC	MIFARE Classic
MIsmartApp	Referring to Delegated Application Management feature (see Section 7.11.2)
MSB	Most Significant Byte
N/A	Not applicable
NAD	Node ADdress, see [6].
NFC	Near Field Communication
NVM	Non-Volatile Memory
OC	Originality Check
OID	Object IDentifier
Over-the-air interface	Interface between PD and TSM.
PC	Proximity Check
PCB	Protocol Control Byte, see [6].
PCD	Proximity Coupling Device (Contactless Reader)
PCDCap2	PCD Capability as exchanged during authentication with AuthenticateEV2First.
PCD-CLF	The Contactless Front End that is part of the PCD.
PCHK	Proximity Check, that is, the whole protocol.
PCSUP	Flag that indicates whether Proximity Check is supported by the PD. The PCSUP is a capability that can in an operator-defined way be transferred in the PDCap1.2 byte, or will be known to the PCD in another way (for example, fixed value for all VCs with the same VCIID).
PD	Proximity Device. Term used to refer to PICC or NFC device in card mode if the distinction between the two is not relevant.
PDCap2	PD Capability as exchanged during authentication with AuthenticateEV2First.
PD-CLF	The Contactless Front End that is part of the PD.
Personalization	The process to bind the PICC to a user, for example, storing person-related information on a PICC used for access management.
PICC	Proximity Integrated Circuit Card (Contactless Card)
PICCKeySettings	The PICC level key configuration settings.
Pre-Personalization	The process to put keys and other Service Operator related data on a PICC at a moment that the user information is not yet known. for example, , configuring PICCs before shipping them to the sales office of the transport operator where it will then be used in the personalization process.
PPS1	Protocol Parameter Selection 1 (PPS1 byte of ([6]))
PRF	Pseudo Random Function
RATS	Request Answer To Select
RC	Return Code

Provided under NDA only.
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-9bfe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 382. Abbreviations and symbols...*continued*

Acronym	Description
Reader device	Reader device or reader is often used to indicate the entire PCD (= Terminal), even though the device also carries out write operations.
ReaderID	Serial number of a reader device
REQA	REQuest command, Type A
Resp	Response
RF	Radio Frequency
RFU	Reserved for Future Use. All bytes that hold this designation must be set to 0x00.
RID	Random ID
RMS	Root Mean Square
RndA	Random number generated by the PCD as part of the native authentication process.
RndB	Random number generated by the PD as part of the native authentication process.
RndC	Random number used as challenge in proximity checking.
RndR	Random number used as response in proximity checking.
SAI	Second Application Indicator
SAK	Select AcKnowledge, see [4]
SAM	Secure Access Module
SE	Secure Element. In this specification used in two forms: UICC and Second Secure Element (SSE).
Secure Messaging	The message exchange with an active authentication
Service Operator	Entity which manages the PCDs and uses them in combination with PDs in the hands of consumers to offer services to consumers. Same as Contactless Infrastructure Operator.
Session	The period between an authentication with a sector key and a subsequent authentication with the same or another sector key, or an error.
SVx	Session vector or intermediate value
Terminal	PCD. See also: Reader device.
TI	Transaction Identifier. A number set after AuthenticateEV2First. The TI is included in the calculation of the MAC and is one of the mechanisms used to ensure that all messages of multiple sessions in a single transaction are cryptographically bound together.
TM	Transaction MAC
TMC	Transaction MAC Counter
TMI	Transaction MAC Input
TMRI	Transaction MAC ReaderID
TMRICur	Transaction MAC ReaderID of the current transaction
TMRIPrev	Transaction MAC ReaderID of the previous transaction
TMV	Transaction MAC Value
UID	Unique IDentifier, see [4]
UseRID	Property of a VC, configured with SetConfiguration, indicating whether or not to use Random ID (RID) in anti-collision for this VC.
VC	Virtual Card, one MIFARE Plus PICC contains one virtual card

Provided under NDA only - Personal copy for: COMPANY CONFIDENTIAL Sergio Salata Guardian Solutions GmbH Document copy, share or reproduce 5fc087791-c22c-495c-9000-6ecc3cae4061

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 382. Abbreviations and symbols...*continued*

Acronym	Description
VCIID	Virtual Card Installation Identifier
VCState	Virtual Card state. The name used for a card state throughout the document.
VCTID	See: Virtual Card Type Identifier
VCTID Override	Parameter that can be used to define the VCTID that the VC will be present instead of the real native VCTID of the VC.
VCUID	Virtual Card UID. Every Virtual Card can have its own UID called VCUID. It uniquely identifies the Virtual Card among all Virtual Cards in the world, like the UID uniquely defines the PD among all PDs in the world.
Virtual Card Installation Identifier	Identifying the installation to which the Virtual Card belongs. A VC belongs to a certain installation if (one of) the IID of the installation is the same as the VCIID.
Virtual Card Type Identifier	Identification of the type of card, for example, MIFARE Plus, MIFARE DESFire, etc. It is a property of every virtual card.
||	concatenated
[n..m]	Bytes m to n of the preceding byte string
[bXX..bYY]	The xx bit of the stream to the yy bit of the stream. 00 is the least significant bit.
[]	Within the command API definition and formulas, the square brackets can have another meaning. They indicate optional parameters.
→	Message flowing from the left-hand entity to the right-hand entity
←	Message flowing from the right-hand entity to the left-hand entity

Provided under NDA by COMPANY CONFIDENTIAL - personal copy for: Sergii Salata Quantag IT Solutions GmbH Do not copy, share or reproduce 5fc0779d-c22c-405c-90fe-6ecc3ce4251

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

14 References

[1] FIPS PUB 197 Federal Information Processing Standards publication - Advanced Encryption Standard (AES), National Institute of Standards and Technology, 2001 November 26.

[2] NIST Special Publication 800-38A - Recommendation for BlockCipher Modes of Operation. http://csrc. nist. gov/publications/nistpubs/800-38a/sp800-38a.pdf

[3] NIST Special Publication 800-38B - Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf.

[4] NIST FIPS PUB 186-4 - Digital Signature Standard (DSS). https://doi.org/10.6028/NIST.FIPS.186-4, July 2013.

[5] ISO/IEC 14443-3:2018 Identification cards – Contactless integrated circuit(s) cards – Proximity cards – Part 3: Initialization and anticollision – Third edition, 07 2018.

[6] ISO/IEC 14443-4:2018 Identification cards – Contactless integrated circuit(s) cards – Proximity cards – Part 4: Transmission protocol – Fourth edition, 07 2018.

[7] ISO/IEC 9594-6:2020 Information technology - Open systems interconnection - Part 6: The Directory: Selected attribute types - Ninth edition, 11 2020.

[8] ISO/IEC 9594-8:2020 Information technology - Open systems interconnection - Part 8: The Directory: Publickey and attribute certificate frameworks - Ninth edition, 11 2020.

[9] ISO/IEC 7812-1:2017 - Identification cards - Identification of issuers - Part 1: Numbering system, 01 2017

[10] NFC Forum - Type 4 Tag - Technical Specification - Version 1.2 - [T4T] - 2021.09.27, 09 2021.

[11] ISO JTC 1/SC 17 - Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange. ISO/IEC 7816-4:2020, May 2020.

[12] ISO JTC 1/SC 27 - Informtion technology - Security techniques - Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher. ISO/IEC 9797-1:1999, 1999.

[13] ISO JTC 1/SC 27 - Information technology – Security techniques – Entity authentication – Part 3: Mechanisms using digital signature techniques. ISO/IEC 9798-3:2019, 2019.

[14] ISO JTC 1/SC 6. Information technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER). ISO/IEC 8825-1:2015, November 2015.

[15] ISO/IEC 61967-4 - Integrated circuits – Measurement of electromagnetic emissions – Part 4: Measurement of conducted emissions – 1 Ω/150 Ω direct coupling method. ISO/IEC 61967-4:2021, March 2021.

[16] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, and David Cooper. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, May 2008.

[17] Bundesamt für Sicherheit in der Informationstechnik, Anwendungshinweise und Interpretationen zum Schema, AIS31: Funktionalitätsklassen und Evaluationsmethodologie für physikalische Zufallszahlengeneratoren, Bundesamt für Sicherheit in der In formationstechnik. Version 2.0, September 18, 2011.

[18] GlobalPlatform, Globalplatform card specification, Version 2.2.1, January 2011.

[19] GlobalPlatform, Globalplatform card - contactless services - card specification v2.2 - admendment c. Version 1.1, April 2013.

[20] GlobalPlatform, Globalplatform technology - apdu transport over spi / I^2C - version 1.0, January 2020.

[21] M. Lochter and J. Merkle. Rfc5639 - Elliptic curve cryptography (ECC) brainpool standard curves and curve generation, March 2010.

[22] G. Lowe - Some new attacks upon security protocols. In Proceedings 9th IEEE Computer Security Foundations
Workshop, pages 162–169, 1996.

[23] National Institute of Standards and Technology (NIST) - Recommendation for key derivation using pseudorandom functions. NIST Special Publication 800-108, November 2008.

Property of NXP Semiconductors
CONFIDENTIAL
Design Data - Solutions Group
Quarterly - Strategic Product
Do not copy, share or reproduce
5fc779dc-22c-a052-2016-9ec5ee42f1

[24] National Institute of Standards and Technology (NIST) - Recommendation for key management part 1: General. NIST Special Publication 800-57, May 2011.

[25] National Institute of Standards and Technology (NIST) - Federal Information Processing Standard (FIPS) 180-4: Secure Hash Standard (SHS). NIST FIPS PUB 180-4, August 2015.

[26] National Institute of Standards and Technology (NIST) - Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography. NIST Special Publication 800-56A, April 2018.

[27] National Institute of Standards and Technology (NIST) - Recommendation for key-derivation methods in keyestablishment schemes. NIST Special Publication 800-56C, August 2020.

[28] Certicom Research, Sec 1: Elliptic curve cryptography, Version 2.0, May 2009.

[29] NXP Semiconductors, Application Note, AN10922: Symmetric key diversifications, March 2010. Referenced document number AN165313.

[30] NXP Semiconductors, Application Note, AN11885: Unique Chip Identification, October 2018. Referenced doc nr: 395111

[31] NXP Semiconductors, User Manual, UM10204 I2C-bus specification and user manual, Rev. 7, 10 2021

[32] IEEE Computer Society. IEEE Standard for Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications. IEEE Std 802.3-2008, December 2008.

[33] Deutsche Kommission Elektrotechnik Elektronik Informationstechnik (VDE-DKE). VDE-AR-E 2532-100 Anwendungsregel: Requirements for an authentication for the use of electric mobility supply systems, September 2020.

[34] P.C. van Oorschot W. Diffie and M.J. Wiener - Authentication and authenticated key exchanges. Designs, Codes and Cryptography 2(2), 107-125, 1992. https://doi.org/10.1007/BF00124891.

[35] R. Housley W. Polk and L. Bassham - Rfc3279: Algorithms and identifiers for the internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile, April 2002.

[36] NXP Semiconductors, Datasheet Addendum, AD9747: Wafer and Delivery Specification

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ec3cae42f1

15 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2024-2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Provided under NDA only
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3f4e4f7b

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

16 Revision history

Table 383. Revision history

Document ID	Release date	Description	Supersedes
MF3E(H)x3 v.1.2	10 July 2025	• Various editorial changes (layout, typos, etc.) • Updated ordering information in Section 4.1: updated MF3EH23A0DA8/01, MF3EH43A0DA8/01, MF3 EH83A0DA8/01, and MF3EH93A0DA8/01 (input capacitance: 17 pf → 70 pf). • Updated ordering information in Section 4.2: updated MF3EH23A0DA8/01EV, MF3EH43A0DA8/01EV, MF3EH83A0DA8/01EV, and MF3EH93A0DA8/01EV (input capacitance: 17 pf → 70 pf). • Updated EV Charging specific Trust Provisioning information for the pre-created application and file configuration in Section 7.19.4.1 (AID parameter, ISO File ID parameter) and Section 7.19.4.3 (ISO File ID parameters) • Updated EV Charging specific details for the EV Charging Certificate content in Section 7.19.4.4 (update of CA ID)	MF3E(H)x3 v.1.1
MF3E(H)x3 v.1.1	8 November 2024	• Various editorial updates (typos; additional details and remarks). • Updated product type list in Section 4. • Updated parameter description and value field for parameter OptsA in Table 78 • Updated description for Option 0x15 in Table 29. • Extended description of reader certificate format in Section 7.10.2. • Updated certificate chain validation in Section 7.10.2.2. • Added details related to the Proximity Check protocol in Section 7.18.2. • Included new chapter related to overall product security in Section 7.23. • Updated SWSubType bit setting description in Table 119. • Included additional details and conditions to Section 9, Section 10 and Section 11.	MF3E(H)x3 v.1.0
MF3E(H)x3 v.1.0	31 July 2024	• Changed Datasheet status to Product Datasheet. • Various editorial updates (typos; additional explanation). • Slightly updated the introduction section in Section 1.1. • Re-organized Section 2. • Updated product type list to reflect SW MinorVersion '01' in Section 4. • Extended product type list to reflect standard product part types and customized product part types in Section 4. • Updated SWSubType bit setting for the customized product configuration in Table 119. • Restructured sections Section 9 for Limiting Values, Section 10 for Recommended Operating Conditions	MF3E(H)x3 v.0.3

Provided under NDA only
COMPANY CONFIDENTIAL
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
Personal copy for:
5f6f7b0c-c22-4b5c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Table 383. Revision history...*continued*

Document ID	Release date	Description	Supersedes
		and Section 11 for Characteristics, to reflect the applicability of given values to the different product package types. • Added additional temperature range values to Limiting Values in Table 368 and Table 370. • Added additional temperature range values to Recommended Operating Conditions in Table 371 and Table 373. • Added additonal values to Electrical AC Characteristics in Table 374 • Included additional product conditions related to temperature range in Section 11.	
MF3E(H)x3 v.0.3	3 July 2024	• Included Pinning Information for MOA8 contactless smartcard module in Section 6. • Included Limiting Values in Section 9. • Included Recommended Operating Conditions in Section 10. • Included Device Characteristics in Section 11.	MF3E(H)x3 v.0.2
MF3E(H)x3 v.0.2	2 July 2024	• Changed Datasheet status to Objective Datasheet. • Various editorial updates (typos; additional explanation). • Added feature to package mapping and explanation of functionality in different delivery types in Section 2.2. • Updated description of I2C interface and GPIOs in Section 2.1.7 and Section 2.1.8. • Updated the description for targeted applications in Section 3. • Updated details related to ordering information in Section 4. • Clarified text to indicate that symmetric key access rights can be obtained through ECC-based authentication in Section 7.8.5.2. • Included Trust Provisioning information for EV Charging in Section 7.19.4. • Adapted response information for GetVersion Part2 response in Table 119. • Updated parameter FabKeyID to TypeID in Section 7.7.2.1 and Table 123.	MF3E(H)x3 v.0.1
MF3E(H)x3 v.0.1	18 December 2023	• First version for external sharing.	-

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-4f0c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Legal information

Data sheet status

Document status[1][2]	Product status[3]	Definition
Objective [short] data sheet	Development	This document contains data from the objective specification for product development.
Preliminary [short] data sheet	Qualification	This document contains data from the preliminary specification.
Product [short] data sheet	Production	This document contains the product specification.

[1] Please consult the most recently issued document before initiating or completing a design.
[2] The term 'short data sheet' is explained in section "Definitions".
[3] The product status of device(s) described in this document may have changed since this document was published and may differ in case of multiple devices. The latest product status information is available on the Internet at URL https://www.nxp.com.

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Short data sheet — A short data sheet is an extract from a full data sheet with the same product type number(s) and title. A short data sheet is intended for quick reference only and should not be relied upon to contain detailed and full information. For detailed and full information see the relevant full data sheet, which is available on request via the local NXP Semiconductors sales office. In case of any inconsistency or conflict with the short data sheet, the full data sheet shall prevail.

Product specification — The information and data provided in a Product data sheet shall define the specification of the product as agreed between NXP Semiconductors and its customer, unless NXP Semiconductors and customer have explicitly agreed otherwise in writing. In no event however, shall an agreement be valid in which the NXP Semiconductors product is deemed to offer functions and qualities beyond those described in the Product data sheet.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

Document ID: 5fd779d-c0a4-4b0e-a3c0-4ff1
CONFIDENTIAL - Personal copy for:
Secure Solutions GmbH
Quantum Technologies
Do not copy, share

Quick reference data — The Quick reference data is an extract of the product data given in the Limiting values and Characteristics sections of this document, and as such is not complete, exhaustive or legally binding.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

MIFARE — is a trademark of NXP B.V.

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Tables

Tab. 1. MF3E(H)x3 available packages and features 7
Tab. 2. Ordering information for MIFARE DUOX 9
Tab. 3. Ordering information for MIFARE DUOX for EV-Charging 10
Tab. 4. Bonding pad assignment for SOT500-4 (MOA8 module) 13
Tab. 5. Notation in formulas and pseudo code 16
Tab. 6. I2C communication interface parameters 20
Tab. 7. Supported communication modes 25
Tab. 8. Overview of expected command communication mode for Secure Messaging 25
Tab. 9. ECC-based mutual authentication 31
Tab. 10. ECC-basedcard-unilateral authentication 37
Tab. 11. Three pass authentication 38
Tab. 12. When to use which authentication command 40
Tab. 13. AES state array byte ordering 45
Tab. 14. PICCData: plain encoding and lengths 51
Tab. 15. PICCDataTag 52
Tab. 16. Access condition values coded on 4 bits 58
Tab. 17. Access condition (coded on 4 bits) mapping to access rights at PICC level 60
Tab. 18. ACMap encoding 60
Tab. 19. CertificateAccess Right encoding 61
Tab. 20. Application access rights, specified via AID 62
Tab. 21. Application access rights, specified via DFName 62
Tab. 22. Delegated Application Management access rights 63
Tab. 23. ISOSelectFile response data 65
Tab. 24. Manufacturer characteristics used as card version 66
Tab. 25. SetConfiguration options list 67
Tab. 26. ProtocolOptions 76
Tab. 27. GPIOxConfig 76
Tab. 28. GPIOxPadCtrl 77
Tab. 29. SetConfiguration key requirements 78
Tab. 30. Supported memory configuration 81
Tab. 31. Supported key types 84
Tab. 32. Keys at PICC level 84
Tab. 33. PICCKeySettings byte definition 88
Tab. 34. Application key set properties 90
Tab. 35. Keys at application level 93
Tab. 36. AppKeySettings byte definition 97
Tab. 37. AppKeySetSettings byte definition 98
Tab. 38. GetKeySettings Key Groups 102
Tab. 39. Comparison: conventional vs. delegated application management 122
Tab. 40. TransactionMAC access rights 140
Tab. 41. Set of Access condition coded on 2 bytes 141
Tab. 42. FileAR.SDMMetaRead values 142
Tab. 43. FileAR.SDMFileRead values 142
Tab. 44. FileAR.SDMFileRead2 values 143
Tab. 45. FileAR.SDMFileRead and FileAR.SDMFileRead2 combinations 143
Tab. 46. Command list associated with access rights 144
Tab. 47. Read GPIO response 185
Tab. 48. EV Charging Certificate 206
Tab. 49. EV Charging Certificate Public Key 207
Tab. 50. PD capability information 209
Tab. 51. PD capabilities during authentication 211
Tab. 52. PCD capabilities during authentication 212
Tab. 53. Overview of native commands, sorted by name 218
Tab. 54. Overview of native response codes 220
Tab. 55. Overview PC related commands, sorted by name 221
Tab. 56. Overview PC related response codes 221
Tab. 57. Overview of ISO7816-4 commands 221
Tab. 58. Overview of ISO7816-4 response codes (SW1|SW2) 222
Tab. 59. Overview of native commands, sorted by command code 223
Tab. 60. Overview of native response codes, sorted by response code 225
Tab. 61. Overview PC related commands, sorted by command code 226
Tab. 62. Overview PC related response codes (sorted by response code) 226
Tab. 63. Overview of ISO7816-4 commands, sorted by command code (INS) 226
Tab. 64. Overview of ISO7816-4 response codes (SW1|SW2) (sorted by response code) 227
Tab. 65. Command description - ISOGeneralAuthenticate 228
Tab. 66. Command parameters description - ISOGeneralAuthenticate 228
Tab. 67. Return code description - ISOGeneralAuthenticate 230
Tab. 68. Response data parameters description - ISOGeneralAuthenticate - ISO9000 230
Tab. 69. Command description - ISOGeneralAuthenticatePart2 231
Tab. 70. Command parameters description - ISOGeneralAuthenticatePart2 231
Tab. 71. Return code description - ISOGeneralAuthenticatePart2 231
Tab. 72. Response data parameters description - ISOGeneralAuthenticatePart2 - ISO9000 233
Tab. 73. Command description - ISOGeneralAuthenticateFinal 234
Tab. 74. Command parameters description - ISOGeneralAuthenticateFinal 234
Tab. 75. Return code description - ISOGeneralAuthenticateFinal 235
Tab. 76. Response data parameters description - ISOGeneralAuthenticateFinal - ISO9000 236

Provided by NXP Semiconductors GmbH
CONFIDENTIAL
Supplied under NDA only
Do not copy, share or reproduce
Quantum Solutions GmbH Personal copy for:
5fc07779d-c22c-4055-b7ff-6ec3ca8241

Tab. 77. Command description - ISOInternalAuthenticate ... 237
Tab. 78. Command parameters description - ISOInternalAuthenticate ... 237
Tab. 79. Return code description - ISOInternalAuthenticate ... 238
Tab. 80. Response data parameters description - ISOInternalAuthenticate - ISO9000 ... 238
Tab. 81. Command description - AuthenticateEV2First ... 239
Tab. 82. Command parameters description - AuthenticateEV2First ... 239
Tab. 83. Return code description - AuthenticateEV2First ... 240
Tab. 84. Response data parameters description - AuthenticateEV2First - ADDITIONAL_FRAME ... 240
Tab. 85. Command description - AuthenticateEV2FirstPart2 ... 240
Tab. 86. Command parameters description - AuthenticateEV2FirstPart2 ... 240
Tab. 87. Return code description - AuthenticateEV2FirstPart2 ... 241
Tab. 88. Response data parameters description - AuthenticateEV2FirstPart2 - OPERATION_OK ... 241
Tab. 89. Command description - AuthenticateEV2NonFirst ... 242
Tab. 90. Command parameters description - AuthenticateEV2NonFirst ... 242
Tab. 91. Return code description - AuthenticateEV2NonFirst ... 243
Tab. 92. Response data parameters description - AuthenticateEV2NonFirst - ADDITIONAL_FRAME ... 243
Tab. 93. Command description - AuthenticateEV2NonFirstPart2 ... 243
Tab. 94. Command parameters description - AuthenticateEV2NonFirstPart2 ... 244
Tab. 95. Return code description - AuthenticateEV2NonFirstPart2 ... 244
Tab. 96. Response data parameters description - AuthenticateEV2NonFirstPart2 - OPERATION_OK ... 244
Tab. 97. Command description - FreeMem ... 245
Tab. 98. Command parameters description - FreeMem ... 245
Tab. 99. Return code description - FreeMem ... 245
Tab. 100. Response data parameters description - FreeMem - OPERATION_OK ... 245
Tab. 101. Command description - Format ... 246
Tab. 102. Command parameters description - Format ... 246
Tab. 103. Return code description - Format ... 246
Tab. 104. Command description - SetConfiguration ... 247
Tab. 105. Command parameters description - SetConfiguration ... 247
Tab. 106. Return code description - SetConfiguration ... 248
Tab. 107. Command description - GetConfiguration ... 252
Tab. 108. Command parameters description - GetConfiguration ... 252
Tab. 109. Return code description - GetConfiguration ... 252
Tab. 110. Response data parameters description - GetConfiguration - OPERATION_OK ... 253
Tab. 111. Response data parameters description - GetConfiguration - OPERATION_OK ... 253
Tab. 112. Command description - GetVersion ... 254
Tab. 113. Command parameters description - GetVersion ... 254
Tab. 114. Return code description - GetVersion ... 254
Tab. 115. Response data parameters description - GetVersion - ADDITIONAL_FRAME ... 255
Tab. 116. Command description - GetVersionPart2 ... 255
Tab. 117. Command parameters description - GetVersionPart2 ... 255
Tab. 118. Return code description - GetVersionPart2 ... 256
Tab. 119. Response data parameters description - GetVersionPart2 - ADDITIONAL_FRAME ... 256
Tab. 120. Command description - GetVersionPart3 ... 256
Tab. 121. Command parameters description - GetVersionPart3 ... 257
Tab. 122. Return code description - GetVersionPart3 ... 257
Tab. 123. Response data parameters description - GetVersionPart3 - OPERATION_OK ... 257
Tab. 124. Command description - GetCardUID ... 258
Tab. 125. Command parameters description - GetCardUID ... 258
Tab. 126. Return code description - GetCardUID ... 258
Tab. 127. Response data parameters description - GetCardUID - OPERATION_OK ... 258
Tab. 128. Command description - ChangeKey ... 259
Tab. 129. Command parameters description - ChangeKey ... 259
Tab. 130. Return code description - ChangeKey ... 260
Tab. 131. Command description - ChangeKeyEV2 ... 263
Tab. 132. Command parameters description - ChangeKeyEV2 ... 263
Tab. 133. Return code description - ChangeKeyEV2 ... 264
Tab. 134. Command description - InitializeKeySet ... 267
Tab. 135. Command parameters description - InitializeKeySet ... 267
Tab. 136. Return code description - InitializeKeySet ... 268
Tab. 137. Command description - FinalizeKeySet ... 269
Tab. 138. Command parameters description - FinalizeKeySet ... 269
Tab. 139. Return code description - FinalizeKeySet ... 269
Tab. 140. Command description - RollKeySet ... 271
Tab. 141. Command parameters description - RollKeySet ... 271
Tab. 142. Return code description - RollKeySet ... 271
Tab. 143. Command description - GetKeySettings ... 273
Tab. 144. Command parameters description - GetKeySettings ... 273
Tab. 145. Return code description - GetKeySettings ... 273
Tab. 146. Response data parameters description - GetKeySettings - OPERATION_OK ... 274
Tab. 147. Response data parameters description - GetKeySettings - OPERATION_OK -

Provided by IHS Markit - Personal copy for: COMPANY SENSITIVE - Data Solutions GmbH, Do not copy, share or reproduce, Customer ID: 5fc0779d-0262-410b-bb0e-a0d1

[Option = 0x01] KeyID.ECCPrivateKey s meta-data ... 275
Tab. 148. Response data parameters description - GetKeySettings - OPERATION_OK - [Option = 0x02] KeyID.CARootKeys meta-data ... 275
Tab. 149. Command description - ChangeKeySettings ... 277
Tab. 150. Command parameters description - ChangeKeySettings ... 277
Tab. 151. Return code description - ChangeKeySettings ... 277
Tab. 152. Command description - GetKeyVersion ... 279
Tab. 153. Command parameters description - GetKeyVersion ... 279
Tab. 154. Return code description - GetKeyVersion ... 280
Tab. 155. Response data parameters description - GetKeyVersion ... 280
Tab. 156. Response data parameters description - GetKeyVersion ... 281
Tab. 157. Command description - ManageKeyPair ... 282
Tab. 158. Command parameters description - ManageKeyPair ... 282
Tab. 159. Return code description - ManageKeyPair ... 284
Tab. 160. Response data parameters description - ManageKeyPair - OPERATION_OK ... 284
Tab. 161. Command description - ManageCARootKey ... 285
Tab. 162. Command parameters description - ManageCARootKey ... 285
Tab. 163. Return code description - ManageCARootKey ... 287
Tab. 164. Command description - ExportKey ... 288
Tab. 165. Command parameters description - ExportKey ... 288
Tab. 166. Return code description - ExportKey ... 288
Tab. 167. Response data parameters description - ExportKey - OPERATION_OK ... 289
Tab. 168. Command description - GetKeySettings ... 290
Tab. 169. Command parameters description - GetKeySettings ... 290
Tab. 170. Return code description - GetKeySettings ... 290
Tab. 171. Response data parameters description - GetKeySettings - OPERATION_OK ... 291
Tab. 172. Response data parameters description - GetKeySettings - OPERATION_OK - [Option = 0x01] KeyID.ECCPrivateKey s meta-data ... 292
Tab. 173. Response data parameters description - GetKeySettings - OPERATION_OK - [Option = 0x02] KeyID.CARootKeys meta-data ... 292
Tab. 174. Command description - CreateApplication ... 294
Tab. 175. Command parameters description - CreateApplication ... 294
Tab. 176. Return code description - CreateApplication ... 296
Tab. 177. Command description - DeleteApplication ... 297
Tab. 178. Command parameters description - DeleteApplication ... 297
Tab. 179. Return code description - DeleteApplication ... 297
Tab. 180. Command description - CreateDelegatedApplication ... 299
Tab. 181. Command parameters description - CreateDelegatedApplication ... 299
Tab. 182. Return code description - CreateDelegatedApplication ... 300
Tab. 183. Command description - CreateDelegatedApplicationPart2 ... 300
Tab. 184. Command parameters description - CreateDelegatedApplicationPart2 ... 300
Tab. 185. Return code description - CreateDelegatedApplicationPart2 ... 301
Tab. 186. Command description - SelectApplication ... 303
Tab. 187. Command parameters description - SelectApplication ... 303
Tab. 188. Return code description - SelectApplication ... 303
Tab. 189. Command description - GetApplicationIDs ... 305
Tab. 190. Command parameters description - GetApplicationIDs ... 305
Tab. 191. Return code description - GetApplicationIDs ... 305
Tab. 192. Response data parameters description - GetApplicationIDs - ADDITIONAL_FRAME ... 306
Tab. 193. Response data parameters description - GetApplicationIDs - OPERATION_OK ... 306
Tab. 194. Command description - GetDFNames ... 307
Tab. 195. Command parameters description - GetDFNames ... 307
Tab. 196. Return code description - GetDFNames ... 307
Tab. 197. Response data parameters description - GetDFNames - ADDITIONAL_FRAME ... 308
Tab. 198. Response data parameters description - GetDFNames - OPERATION_OK ... 308
Tab. 199. Command description - GetDelegatedInfo ... 309
Tab. 200. Command parameters description - GetDelegatedInfo ... 309
Tab. 201. Return code description - GetDelegatedInfo ... 309
Tab. 202. Response data parameters description - GetDelegatedInfo - OPERATION_OK ... 310
Tab. 203. Command description - CreateStdDataFile ... 311
Tab. 204. Command parameters description - CreateStdDataFile ... 311
Tab. 205. Return code description - CreateStdDataFile ... 312
Tab. 206. Command description - CreateBackupDataFile ... 314
Tab. 207. Command parameters description - CreateBackupDataFile ... 314
Tab. 208. Return code description - CreateBackupDataFile ... 315
Tab. 209. Command description - CreateValueFile ... 316
Tab. 210. Command parameters description - CreateValueFile ... 316
Tab. 211. Return code description - CreateValueFile ... 317
Tab. 212. Command description - CreateLinearRecordFile ... 319

Provided by IHS Markup to: COMPANY CONFIDENTIAL - Personal copy for: Quarta-Rys Solutions GmbH Do not copy, share or reproduce. 5fc0779d-c22c-4a0f-b0ee-c4e4f

Tab. 213. Command parameters description - CreateLinearRecordFile ... 319
Tab. 214. Return code description - CreateLinearRecordFile ... 320
Tab. 215. Command description - CreateCyclicRecordFile ... 321
Tab. 216. Command parameters description - CreateCyclicRecordFile ... 321
Tab. 217. Return code description - CreateCyclicRecordFile ... 322
Tab. 218. Command description - CreateTransactionMACFile ... 323
Tab. 219. Command parameters description - CreateTransactionMACFile ... 323
Tab. 220. Return code description - CreateTransactionMACFile ... 324
Tab. 221. Command description - DeleteFile ... 326
Tab. 222. Command parameters description - DeleteFile ... 326
Tab. 223. Return code description - DeleteFile ... 326
Tab. 224. Command description - GetFileIDs ... 328
Tab. 225. Command parameters description - GetFileIDs ... 328
Tab. 226. Return code description - GetFileIDs ... 328
Tab. 227. Response data parameters description - GetFileIDs - OPERATION_OK ... 328
Tab. 228. Command description - GetISOFileIDs ... 329
Tab. 229. Command parameters description - GetISOFileIDs ... 329
Tab. 230. Return code description - GetISOFileIDs ... 329
Tab. 231. Response data parameters description - GetISOFileIDs - ADDITIONAL_FRAME ... 330
Tab. 232. Response data parameters description - GetISOFileIDs - OPERATION_OK ... 330
Tab. 233. Command description - GetFileSettings ... 331
Tab. 234. Command parameters description - GetFileSettings ... 331
Tab. 235. Return code description - GetFileSettings ... 332
Tab. 236. Response data parameters description - GetFileSettings - Targeting FileType.StandardData or FileType.BackupData ... 332
Tab. 237. Response data parameters description - GetFileSettings - Targeting CRLFile ... 333
Tab. 238. Response data parameters description - GetFileSettings - Targeting FileType.Value ... 334
Tab. 239. Response data parameters description - GetFileSettings - Targeting FileType.LinearRecord or FileType.CyclicRecord ... 335
Tab. 240. Response data parameters description - GetFileSettings - Targeting FileType.TransactionMAC ... 336
Tab. 241. Command description - GetFileCounters ... 338
Tab. 242. Command parameters description - GetFileCounters ... 338
Tab. 243. Return code description - GetFileCounters ... 339
Tab. 244. Response data parameters description - GetFileCounters - OPERATION_OK - Targeting FileType.StandardData with SDM enabled ... 339
Tab. 245. Command description - ChangeFileSettings ... 340
Tab. 246. Command parameters description - ChangeFileSettings ... 340
Tab. 247. Return code description - ChangeFileSettings ... 344
Tab. 248. Command description - ReadData ... 349
Tab. 249. Command parameters description - ReadData ... 349
Tab. 250. Return code description - ReadData ... 350
Tab. 251. Response data parameters description - ReadData - ADDITIONAL_FRAME ... 351
Tab. 252. Response data parameters description - ReadData - OPERATION_OK ... 351
Tab. 253. Response data parameters description - ReadData - OPERATION_OK - Targeting CRLFile with Offset = 0xFFFFFF ... 351
Tab. 254. Command description - WriteData ... 352
Tab. 255. Command parameters description - WriteData ... 352
Tab. 256. Return code description - WriteData ... 353
Tab. 257. Command description - GetValue ... 355
Tab. 258. Command parameters description - GetValue ... 355
Tab. 259. Return code description - GetValue ... 355
Tab. 260. Response data parameters description - GetValue - OPERATION_OK ... 356
Tab. 261. Command description - Credit ... 357
Tab. 262. Command parameters description - Credit ... 357
Tab. 263. Return code description - Credit ... 357
Tab. 264. Command description - Debit ... 359
Tab. 265. Command parameters description - Debit ... 359
Tab. 266. Return code description - Debit ... 359
Tab. 267. Command description - LimitedCredit ... 361
Tab. 268. Command parameters description - LimitedCredit ... 361
Tab. 269. Return code description - LimitedCredit ... 361
Tab. 270. Command description - ReadRecords ... 363
Tab. 271. Command parameters description - ReadRecords ... 363
Tab. 272. Return code description - ReadRecords ... 364
Tab. 273. Response data parameters description - ReadRecords - ADDITIONAL_FRAME ... 365
Tab. 274. Response data parameters description - ReadRecords - OPERATION_OK ... 365
Tab. 275. Command description - WriteRecord ... 366
Tab. 276. Command parameters description - WriteRecord ... 366
Tab. 277. Return code description - WriteRecord ... 367
Tab. 278. Command description - UpdateRecord ... 368
Tab. 279. Command parameters description - UpdateRecord ... 368
Tab. 280. Return code description - UpdateRecord ... 369
Tab. 281. Command description - ClearRecordFile ... 371
Tab. 282. Command parameters description - ClearRecordFile ... 371
Tab. 283. Return code description - ClearRecordFile ... 371

Provided by NXP Semiconductors
COMPANY CONFIDENTIAL - Personal copy for:
Content Solutions CmPH
Quoting TTS
Do not copy, share or reproduce
5fc0779d-02c-410c-bb0e-ec3e4f

Tab. 284. Command description - CommitTransaction ..373
Tab. 285. Command parameters description - CommitTransaction 373
Tab. 286. Return code description - CommitTransaction 374
Tab. 287. Response data parameters description - CommitTransaction - OPERATION_OK 374
Tab. 288. Response data parameters description - CommitTransaction - OPERATION_OK 374
Tab. 289. Command description - AbortTransaction 375
Tab. 290. Command parameters description - AbortTransaction 375
Tab. 291. Return code description - AbortTransaction .. 375
Tab. 292. Command description - CommitReaderID 376
Tab. 293. Command parameters description - CommitReaderID 376
Tab. 294. Return code description - CommitReaderID ..376
Tab. 295. Response data parameters description - CommitReaderID - OPERATION_OK 377
Tab. 296. Command description - CryptoRequest 378
Tab. 297. Command parameters description - CryptoRequest 378
Tab. 298. Return code description - CryptoRequest 378
Tab. 299. Response data parameters description - CryptoRequest - OPERATION_OK 379
Tab. 300. Command description - CryptoRequest_ECCSign 380
Tab. 301. Command parameters description - CryptoRequest_ECCSign 380
Tab. 302. Return code description - CryptoRequest_ECCSign 381
Tab. 303. Response data parameters description - CryptoRequest_ECCSign - OPERATION_OK 381
Tab. 304. Command description - CryptoRequest_Echo 382
Tab. 305. Command parameters description - CryptoRequest_Echo 382
Tab. 306. Return code description - CryptoRequest_Echo 382
Tab. 307. Response data parameters description - CryptoRequest_Echo - OPERATION_OK 382
Tab. 308. Command description - ManageGPIO 383
Tab. 309. Command parameters description - ManageGPIO 383
Tab. 310. Return code description - ManageGPIO 385
Tab. 311. Response data parameters description - ManageGPIO - OPERATION_OK 386
Tab. 312. Response data parameters description - ManageGPIO - OPERATION_OK 386
Tab. 313. Command description - ReadGPIO 387
Tab. 314. Command parameters description - ReadGPIO 387
Tab. 315. Return code description - ReadGPIO 387
Tab. 316. Response data parameters description - ReadGPIO - OPERATION_OK - GPIOStatus bytes as defined in 388
Tab. 317. Command description - ISOSelectFile 389
Tab. 318. Command parameters description - ISOSelectFile 389
Tab. 319. Return code description - ISOSelectFile 390
Tab. 320. Response data parameters description - ISOSelectFile - ISO9000 391
Tab. 321. Response data parameters description - ISOSelectFile - ISO9000 391
Tab. 322. Response data parameters description - ISOSelectFile - ISO9000 391
Tab. 323. Response data parameters description - ISOSelectFile - ISO6283 - Application selected with limited functionality: TMCLimit reached 391
Tab. 324. Response data parameters description - ISOSelectFile - ISO6283 - Application selected with limited functionality: KeyID.AppTransactionSIGKey does not exist or is not enabled for Transaction Signature 391
Tab. 325. Response data parameters description - ISOSelectFile - ISO6283 - Application selected with limited functionality: KeyID.AppTransactionSIGKey has enabled KeyUsageCtrLimit reached 392
Tab. 326. Command description - ISOReadBinary 393
Tab. 327. Command parameters description - ISOReadBinary 393
Tab. 328. Return code description - ISOReadBinary 394
Tab. 329. Response data parameters description - ISOReadBinary - ISO9000 394
Tab. 330. Command description - ISOUpdateBinary 395
Tab. 331. Command parameters description - ISOUpdateBinary 395
Tab. 332. Return code description - ISOUpdateBinary ..396
Tab. 333. Command description - ISOReadRecord 398
Tab. 334. Command parameters description - ISOReadRecord 398
Tab. 335. Return code description - ISOReadRecord ... 399
Tab. 336. Response data parameters description - ISOReadRecord - ISO9000 399
Tab. 337. Command description - ISOAppendRecord .. 400
Tab. 338. Command parameters description - ISOAppendRecord 400
Tab. 339. Return code description - ISOAppendRecord 400
Tab. 340. Command description - ISOGetChallenge 402
Tab. 341. Command parameters description - ISOGetChallenge 402
Tab. 342. Return code description - ISOGetChallenge . 402
Tab. 343. Response data parameters description - ISOGetChallenge - ISO9000 402
Tab. 344. Command description - PreparePC 403
Tab. 345. Command parameters description - PreparePC 403
Tab. 346. Return code description - PreparePC 403
Tab. 347. Response data parameters description - PreparePC - OK 404
Tab. 348. Response data parameters description - PreparePC - OK 404

Provided by NXP CONFIDENTIAL - Personal copy for: Copyright SPACE-IT Solutions GmbH Do not copy, share or repeat 5fc0779d-22c4-4f5-9ef6-ec8c8421

Tab. 349. Command description - ProximityCheck 405
Tab. 350. Command parameters description - ProximityCheck 405
Tab. 351. Return code description - ProximityCheck 405
Tab. 352. Response data parameters description - ProximityCheck - OK_PC 406
Tab. 353. Command description - VerifyPC 407
Tab. 354. Command parameters description - VerifyPC 407
Tab. 355. Return code description - VerifyPC 407
Tab. 356. Response data parameters description - VerifyPC - OK 407
Tab. 357. Command description - VDE_ReadData 408
Tab. 358. Command parameters description - VDE_ReadData 408
Tab. 359. Return code description - VDE_ReadData 408
Tab. 360. Response data parameters description - VDE_ReadData - ISO9000 409
Tab. 361. Command description - VDE_WriteData 410
Tab. 362. Command parameters description - VDE_WriteData 410
Tab. 363. Return code description - VDE_WriteData 410
Tab. 364. Command description - VDE_ECDSASign 412
Tab. 365. Command parameters description - VDE_ECDSASign 412
Tab. 366. Return code description - VDE_ECDSASign 412
Tab. 367. Response data parameters description - VDE_ECDSASign - ISO9000 413
Tab. 368. Limiting values for MOA8 415
Tab. 369. Limiting values for bumped wafer on FFC 415
Tab. 370. Limiting values for WLCSP and HVQFN20 ... 415
Tab. 371. Recommended operating conditions for MOA8 417
Tab. 372. Recommended operating conditions for bumped wafer on FFC 417
Tab. 373. Recommended operating conditions for WLCSP and HVQFN20 417
Tab. 374. Electrical AC Characteristics of antenna pins LA, LB 418
Tab. 375. Non-Volatile memory timing characteristics ...418
Tab. 376. Electrical AC Characteristics of SDA, SCL 419
Tab. 377. Electrical DC characteristics of GPIO1 419
Tab. 378. Eletrical DC characteristics of GPIO1/2 420
Tab. 379. Electrical DC characteristics of I2C 422
Tab. 380. Electrical DC characteristics of IC supply voltage VCC 423
Tab. 381. MF3E(H)x3 package outline for MOA8 delivery 425
Tab. 382. Abbreviations and symbols 426
Tab. 383. Revision history 434

Provided under NDA only.
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Figures

Fig. 1. NFC Forum certification for MF3E(H)x3 2
Fig. 2. Block diagram 12
Fig. 3. Contact assignments for SOT500-4 (MOA8 module) 13
Fig. 4. Native command format 22
Fig. 5. Authentication State Diagram 28
Fig. 6. Session key generation for EV2 Secure Messaging 42
Fig. 7. EV2 Secure Messaging: MAC protection mode 48
Fig. 8. EV2 Secure Messaging: Full protection mode 49
Fig. 9. Secure Dynamic Messaging for Reading example 57
Fig. 10. Access Conditions Example 59
Fig. 11. Key architecture 83
Fig. 12. MF3E(H)x3 application key architecture 89
Fig. 13. Application key set life cycle state diagram 92
Fig. 14. Certificate file content 110
Fig. 15. CRLFile format 118
Fig. 16. Transaction MAC file content 140
Fig. 17. MIFARE DESFire file access rights associated with access conditions grouped in sets 141
Fig. 18. Transaction MAC state diagram 171
Fig. 19. NFC Pause Example 183
Fig. 20. NFC Pause Example with ISOReadBinary 183
Fig. 21. ISO/IEC 7816-4 command format 187
Fig. 22. ISO/IEC 7816-4 wrapping command format 188
Fig. 23. Proximity Check set-up 196
Fig. 24. Proximity Check messages 197
Fig. 25. ISOGeneralAuthenticate command protocol 228
Fig. 26. ISOGeneralAuthenticateFinal command protocol 234
Fig. 27. ISOInternalAuthenticate command protocol 237
Fig. 28. AuthenticateEV2First command protocol 239
Fig. 29. AuthenticateEV2NonFirst command protocol 242
Fig. 30. FreeMem command protocol 245
Fig. 31. Format command protocol 246
Fig. 32. SetConfiguration command protocol 247
Fig. 33. GetConfiguration command protocol 252
Fig. 34. GetVersion command protocol 254
Fig. 35. GetCardUID command protocol 258
Fig. 36. ChangeKey command protocol 259
Fig. 37. ChangeKeyEV2 command protocol 263
Fig. 38. InitializeKeySet command protocol 267
Fig. 39. FinalizeKeySet command protocol 269
Fig. 40. RollKeySet command protocol 271
Fig. 41. GetKeySettings command protocol 273
Fig. 42. ChangeKeySettings command protocol 277
Fig. 43. GetKeyVersion command protocol 279
Fig. 44. ManageKeyPair command protocol 282
Fig. 45. ManageCARootKey command protocol 285
Fig. 46. ExportKey command protocol 288
Fig. 47. GetKeySettings command protocol 290
Fig. 48. CreateApplication command protocol 294
Fig. 49. DeleteApplication command protocol 297
Fig. 50. CreateDelegatedApplication command protocol 299
Fig. 51. SelectApplication command protocol 303
Fig. 52. GetApplicationIDs command protocol 305
Fig. 53. GetDFNames command protocol 307
Fig. 54. GetDelegatedInfo command protocol 309
Fig. 55. CreateStdDataFile command protocol 311
Fig. 56. CreateBackupDataFile command protocol 314
Fig. 57. CreateValueFile command protocol 316
Fig. 58. CreateLinearRecordFile command protocol 319
Fig. 59. CreateCyclicRecordFile command protocol 321
Fig. 60. CreateTransactionMACFile command protocol 323
Fig. 61. DeleteFile command protocol 326
Fig. 62. GetFileIDs command protocol 328
Fig. 63. GetISOFileIDs command protocol 329
Fig. 64. GetFileSettings command protocol 331
Fig. 65. GetFileCounters command protocol 338
Fig. 66. ChangeFileSettings command protocol 340
Fig. 67. ReadData command protocol 349
Fig. 68. WriteData command protocol 352
Fig. 69. GetValue command protocol 355
Fig. 70. Credit command protocol 357
Fig. 71. Debit command protocol 359
Fig. 72. LimitedCredit command protocol 361
Fig. 73. ReadRecords command protocol 363
Fig. 74. WriteRecord command protocol 366
Fig. 75. UpdateRecord command protocol 368
Fig. 76. ClearRecordFile command protocol 371
Fig. 77. CommitTransaction command protocol 373
Fig. 78. AbortTransaction command protocol 375
Fig. 79. CommitReaderID command protocol 376
Fig. 80. CryptoRequest command protocol 378
Fig. 81. CryptoRequest_ECCSign command protocol 380
Fig. 82. CryptoRequest_Echo command protocol 382
Fig. 83. ManageGPIO command protocol 383
Fig. 84. ReadGPIO command protocol 387
Fig. 85. ISOSelectFile command protocol 389
Fig. 86. ISOReadBinary command protocol 393
Fig. 87. ISOUpdateBinary command protocol 395
Fig. 88. ISOReadRecord command protocol 398
Fig. 89. ISOAppendRecord command protocol 400
Fig. 90. ISOGetChallenge command protocol 402
Fig. 91. PreparePC command protocol 403
Fig. 92. ProximityCheck command protocol 405
Fig. 93. VerifyPC command protocol 407
Fig. 94. VDE_ReadData command protocol 408
Fig. 95. VDE_WriteData command protocol 410
Fig. 96. VDE_ECDSASign command protocol 412
Fig. 97. Input characteristics of GPIO1/2 422
Fig. 98. MF3E(H)x3 package outline for bumped sawn wafer on FFC delivery 424

Provided under NDA ONLY
CONFIDENTIAL Personal copy for:
Secure Data Solutions Canon
Do not copy, share or reproduce
5fc7c79d-22c2-405e-8f6e-6e63e8ad4

Fig. 99. Package outline SOT500-4 (MOA8) 425

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Sergii Salata
Quantag IT Solutions GmbH
Do not copy, share or reproduce
5fc0779d-c22c-405c-90fe-6ecc3cae42f1

All information provided in this document is subject to legal disclaimers.

© 2025 NXP B.V. All rights reserved.

Contents

1 General description ... 1
1.1 Introduction ... 1
2 Features and benefits ... 3
2.1 Feature overview ... 3
2.1.1 RF interface: ISO/IEC 14443 Type A ... 3
2.1.2 Non-volatile memory ... 3
2.1.3 NV-memory organization and multi-application support ... 3
2.1.4 Security and privacy ... 3
2.1.5 ISO/IEC 7816 compatibility ... 4
2.1.6 Special features ... 5
2.1.7 I2C interface ... 6
2.1.8 GPIO ... 6
2.2 Feature to package mapping ... 7
3 Target applications ... 8
4 Ordering information ... 9
4.1 Ordering information for MIFARE DUOX MF3E(H)x3 product ... 9
4.2 Ordering information for MIFARE DUOX EV-Charging MF3E(H)x3../01EV product ... 10
5 Block diagram ... 12
6 Pinning information ... 13
7 Functional description ... 14
7.1 Introduction ... 14
7.2 Terminology ... 16
7.2.1 Normative statements and informative comment ... 16
7.2.2 Notation in formulas and pseudo code ... 16
7.3 ISO/IEC 14443 support ... 19
7.3.1 Default parameter values ... 19
7.3.1.1 ATQA ... 19
7.3.1.2 SAK ... 19
7.3.1.3 ATS ... 19
7.3.1.4 UID ... 19
7.4 I2C support ... 20
7.4.1 Default I2C parameter values ... 20
7.4.1.1 Target address ... 20
7.4.1.2 Communication interface parameters ... 20
7.4.2 I2C Application Remarks ... 21
7.4.2.1 Power Management ... 21
7.4.2.2 Write after Write behavior ... 21
7.4.2.3 Waiting Time Extension behavior ... 21
7.5 Authentication and Secure Messaging ... 22
7.5.1 Native Command Format ... 22
7.5.1.1 Byte Order ... 22
7.5.2 Command Chaining and Multiple Pass Commands ... 23
7.5.2.1 ISO/IEC 14443-4 Chaining ... 23
7.5.2.2 Native Chaining ... 23
7.5.3 Error Handling ... 24
7.5.4 Communication Modes ... 25
7.5.5 State Diagram ... 27
7.5.6 ECC-based Mutual and Reader-Unilateral Authentication ... 29
7.5.6.1 Data structures and notations ... 29
7.5.6.2 Cryptographic primitives ... 30
7.5.6.3 Authentication overview ... 31
7.5.6.4 Session Key Generation ... 33
7.5.6.5 ISOGeneralAuthenticate and ISOGeneralAuthenticatePart2 ... 33
7.5.6.6 ISOGeneralAuthenticateFinal initiated by ISOSelectFile ... 36
7.5.7 ECC-based Card-Unilateral Authentication ... 36
7.5.7.1 Data structures and notations ... 36
7.5.7.2 Cryptographic primitives ... 37
7.5.7.3 Authentication overview ... 37
7.5.7.4 ISOInternalAuthenticate ... 38
7.5.8 AES-based Symmetric Authentication ... 38
7.5.8.1 Authentication overview ... 38
7.5.8.2 AuthenticateEV2First Command ... 40
7.5.8.3 AuthenticateEV2NonFirst Command ... 41
7.5.8.4 Session Key Generation ... 41
7.5.9 Authentication Counter and Limit ... 43
7.5.10 EV2 Secure Messaging ... 44
7.5.10.1 Transaction Identifier ... 44
7.5.10.2 Command Counter ... 44
7.5.10.3 MAC Calculation ... 45
7.5.10.4 Encryption ... 46
7.5.10.5 Session Key Generation ... 46
7.5.10.6 Plain Communication Mode ... 47
7.5.10.7 MAC Communication Mode ... 47
7.5.10.8 Full Communication Mode ... 48
7.5.11 Secure Dynamic Messaging ... 49
7.5.11.1 SDM for Reading ... 50
7.6 Access Rights Management ... 57
7.6.1 Access conditions ... 58
7.6.2 PICC-level access rights ... 59
7.6.3 KeyID.CARootKey access rights ... 60
7.6.4 Certificate access rights ... 61
7.6.4.1 PICC/Application level access rights ... 62
7.6.4.2 Delegated Application Management access rights ... 63
7.7 Card Memory and Configuration Management ... 64
7.7.1 Card UID ... 64
7.7.1.1 Random ID ... 64
7.7.1.2 GetCardUID Command ... 64
7.7.1.3 Fast confidential UID retrieval with ISOSelectFile ... 64
7.7.2 Card Version ... 66
7.7.2.1 GetVersion Command ... 67
7.7.3 Card Configuration ... 67
7.7.3.1 SetConfiguration Command ... 67
7.7.3.2 GetConfiguration Command ... 81
7.7.4 Memory Management ... 81
7.7.4.1 Free memory with FreeMem command ... 82
7.7.4.2 Format memory with Format command ... 82
7.8 Symmetric Key management ... 83
7.8.1 Key Hierarchy ... 83
7.8.1.1 Key Protection ... 83
7.8.1.2 Session Keys ... 83
7.8.2 Key Types ... 84

Provided under NDA only
CONFIDENTIAL Personal copy for:
Copyright Data Security Consulting GmbH
Do not copy, share or reproduce
5f0759d-322c-4095-8e6-3cae2A1

7.8.3 Key Versioning 84
7.8.4 PICC Level 84
7.8.4.1 PICC Keys 84
7.8.4.2 PICCKeySettings 88
7.8.5 Application Level 88
7.8.5.1 Key Sets 89
7.8.5.2 Application Keys 93
7.8.5.3 Application Key Settings 97
7.8.6 Key Management Commands 98
7.8.6.1 ChangeKey Command 98
7.8.6.2 ChangeKeyEV2 Command 100
7.8.6.3 InitializeKeySet Command 100
7.8.6.4 FinalizeKeySet Command 100
7.8.6.5 RollKeySet Command 101
7.8.6.6 GetKeySettings Command 101
7.8.6.7 ChangeKeySettings Command 102
7.8.6.8 GetKeyVersion Command 102
7.8.7 Memory Consumption 103
7.9 Asymmetric Key management 103
7.9.1 KeyID.ECCPrivateKey Management 104
7.9.1.1 ManageKeyPair command 104
7.9.1.2 KeyID.ECCPrivateKey Key Usage Limit 104
7.9.1.3 KeyID.ECCPrivateKey Information Retrieval 105
7.9.2 KeyID.CARootKey Management 105
7.9.2.1 ManageCARootKey command 105
7.9.2.2 KeyID.CARootKey Information Retrieval 106
7.9.3 ExportKey command 106
7.9.4 PICC Level 106
7.9.4.1 KeyID.ECCPrivateKey entries 106
7.9.4.2 KeyID.CARootKey entries 107
7.9.5 Application level 107
7.9.5.1 KeyID.ECCPrivateKey entries 107
7.9.5.2 KeyID.CARootKey entries 107
7.9.5.3 Initial trust in ECC-only applications 107
7.9.6 Memory Consumption 109
7.10 Certificate management 109
7.10.1 Card Certificates 109
7.10.1.1 Certificate Format and Storage 109
7.10.1.2 PICC Level 110
7.10.1.3 Application Level 111
7.10.1.4 Trust Provisioned Certificates 111
7.10.2 Reader Certificates 112
7.10.2.1 Certificate Format 112
7.10.2.2 Certificate Chain Validation 114
7.10.2.3 Certificate Revocation 116
7.11 Application management 122
7.11.1 Conventional Application Management 123
7.11.1.1 CreateApplication Command 123
7.11.1.2 DeleteApplication Command 124
7.11.2 Delegated Application Management 125
7.11.2.1 Introduction 125
7.11.2.2 CreateDelegatedApplication Command 128
7.11.2.3 Delegated application deletion 132
7.11.2.4 Quota Limit Handling 132
7.11.3 Application selection with SelectApplication command 133
7.11.3.1 Shared Application Management 133
7.11.3.2 SelectApplication Command 134
7.11.4 Application Information Retrieval 134
7.11.4.1 GetApplicationIDs Command 134
7.11.4.2 GetDFNames Command 135
7.11.4.3 GetDelegatedInfo Command 135
7.11.5 Memory Consumption 136
7.12 File management 137
7.12.1 File Types 137
7.12.1.1 FileType.StandardData 137
7.12.1.2 FileType.BackupData 137
7.12.1.3 FileType.Value 138
7.12.1.4 Linear and Cyclic Record Files 138
7.12.1.5 FileType.TransactionMAC 139
7.12.2 File Access Rights Management 140
7.12.2.1 Secure Dynamic Messaging related access rights 142
7.12.2.2 Access right association with commands 144
7.12.2.3 ChangeFileSettings Command 145
7.12.3 File Information Retrieval 146
7.12.3.1 GetFileSettings Command 146
7.12.3.2 GetFileCounters Command 146
7.12.3.3 GetFileIDs Command 147
7.12.3.4 GetISOFileIDs Command 147
7.12.4 File Creation 148
7.12.4.1 CreateStdDataFile Command 148
7.12.4.2 CreateBackupDataFile Command 149
7.12.4.3 CreateValueFile Command 149
7.12.4.4 CreateLinearRecordFile Command 149
7.12.4.5 CreateCyclicRecordFile Command 149
7.12.4.6 CreateTransactionMACFile Command 150
7.12.5 File deletion with DeleteFile command 150
7.12.6 Memory Consumption 151
7.13 Data management 152
7.13.1 Standard and Backup data files 152
7.13.1.1 ReadData Command 152
7.13.1.2 WriteData Command 153
7.13.2 Value files 154
7.13.2.1 GetValue command 154
7.13.2.2 Credit command 155
7.13.2.3 Debit command 156
7.13.2.4 LimitedCredit command 157
7.13.3 Linear and Cyclic Record Files 158
7.13.3.1 ReadRecords Command 158
7.13.3.2 WriteRecord Command 159
7.13.3.3 UpdateRecord Command 160
7.13.3.4 ClearRecordFile Command 161
7.13.4 Transaction MAC files 162
7.13.4.1 ReadData Command 162
7.14 Transaction management 163
7.14.1 Transaction initiation 163
7.14.2 Transaction conclusion 163
7.14.2.1 CommitTransaction command 163
7.14.2.2 AbortTransaction command 165
7.14.3 Transaction MAC 165
7.14.3.1 General overview 166
7.14.3.2 Cryptographic and other concepts 167
7.14.3.3 Transaction MAC enabling 169
7.14.3.4 Transaction MAC calculation 170
7.14.3.5 Transaction MAC and ISO/IEC 7816-4 174
7.14.4 Transaction Signature 174

Provided by IHS Markit under license with NXP Semiconductors USA, Inc.
COMPANY CONFIDENTIAL - Personal copy for:
Data Management Solutions Group
Do not copy, share or reproduce
5fc07f9d-2024-10-15 Sat 02:21

7.14.4.1 General overview ... 175
7.14.4.2 Cryptographic and other concepts ... 175
7.14.4.3 Transaction Signature enabling ... 176
7.14.4.4 Transaction Signature calculation ... 176
7.14.4.5 Transaction Signature and ISO/IEC 7816-4 ... 177
7.14.4.6 Authentication vs Transaction MAC/ Signature ... 178
7.14.5 Transaction Timer ... 178
7.15 Crypto API ... 180
7.15.1 ECCSign ... 180
7.15.2 Echo ... 180
7.16 GPIO management ... 181
7.16.1 NFC Pause feature ... 181
7.16.2 Manage GPIO ... 184
7.16.3 Read GPIO ... 184
7.16.4 Mirroring in the NDEF message ... 185
7.16.5 Authentication notification ... 186
7.16.6 NFC field notification ... 186
7.17 ISO/IEC 7816-4 Support ... 187
7.17.1 ISO/IEC 7816-4 command format ... 187
7.17.2 Wrapping of Native APDUs ... 188
7.17.3 Standard ISO/IEC 7816-4 commands ... 189
7.17.3.1 Byte order ... 189
7.17.3.2 Security Concept of ISO/IEC 7816-4 commands ... 189
7.17.3.3 Error handling ... 189
7.17.3.4 ISOSelectFile command ... 190
7.17.3.5 ISOReadBinary command ... 191
7.17.3.6 ISOUpdateBinary command ... 192
7.17.3.7 ISOReadRecord command ... 192
7.17.3.8 ISOAppendRecord command ... 193
7.17.3.9 ISOGetChallenge command ... 194
7.17.4 Selection of native frame versus ISO/IEC 7816-4 framing ... 194
7.18 Proximity Check Architecture ... 195
7.18.1 Technological basis for proximity checking ... 195
7.18.2 Proximity Check protocol overview ... 196
7.18.2.1 PreparePC command ... 198
7.18.2.2 ProximityCheck command ... 199
7.18.2.3 VerifyPC command ... 200
7.18.3 ISO/IEC 7816-4 support ... 200
7.18.4 Proximity Check location and keys ... 200
7.19 EV Charging ... 202
7.19.1 Enabling EV Charging support ... 202
7.19.2 Data structures ... 202
7.19.3 Command support ... 203
7.19.3.1 VDE_ReadData ... 203
7.19.3.2 VDE_WriteData ... 203
7.19.3.3 VDE_ECDSASign ... 204
7.19.4 Trust Provisioning of EV Charging Data Structures and Key Material ... 204
7.19.4.1 EV Charging application ... 204
7.19.4.2 EV Charging KeyID,ECCPrivateKey entry ... 205
7.19.4.3 EV Charging Files ... 205
7.19.4.4 EV Charging Certificate ... 206
7.20 Upgradeability architecture ... 208
7.20.1 Capability negotiation ... 208
7.20.2 Upgrading the MIFARE DESFire EV2 Secure Messaging Protocol ... 208
7.20.3 Upgrading the ECC-based mutual and reader-unilateral authentication ... 208
7.20.4 Coding of capabilities and detail behavior ... 209
7.20.4.1 Capability exchange during virtual card selection ... 209
7.20.4.2 Capability exchange during symmetric authentication ... 211
7.21 Originality Check architecture ... 214
7.21.1 Dynamic Originality Check Procedure using ECC ... 214
7.21.1.1 Originality key pair ... 214
7.21.1.2 Originality certificate ... 214
7.21.1.3 Card-unilateral authentication ... 214
7.22 Trust Provisioning ... 215
7.22.1 Originality check key pair and certificate ... 215
7.22.2 PICC functionality key pair and certificate ... 215
7.22.3 NXP Delegated Application Management keys ... 215
7.22.4 Commercial customization options ... 215
7.23 Security ... 216
7.23.1 Introduction ... 216
7.23.2 Reset ... 216
7.23.3 Sensor Architecture ... 216
7.23.4 Scalable Security ... 217
8 Commands ... 218
8.1 Introduction ... 218
8.1.1 Overview tables (sorted by name) ... 218
8.1.1.1 Native commands (sorted by name) ... 218
8.1.1.2 PC commands (sorted by name) ... 221
8.1.1.3 ISO7816-4 standard commands (sorted by name) ... 221
8.1.2 Overview tables (sorted by respond/ command code) ... 223
8.1.2.1 Native commands (sorted by command code) ... 223
8.1.2.2 PC commands (sorted by command code) ... 226
8.1.2.3 ISO7816-4 standard commands (sorted by command code) ... 226
8.2 Authentication and secure messaging ... 228
8.2.1 ISOGeneralAuthenticate ... 228
8.2.2 ISOGeneralAuthenticateFinal ... 234
8.2.3 ISOInternalAuthenticate ... 237
8.2.4 AuthenticateEV2First ... 239
8.2.5 AuthenticateEV2NonFirst ... 242
8.3 Memory and configuration management ... 245
8.3.1 FreeMem ... 245
8.3.2 Format ... 246
8.3.3 SetConfiguration ... 247
8.3.4 GetConfiguration ... 252
8.3.5 GetVersion ... 254
8.3.6 GetCardUID ... 258
8.4 Symmetric key management ... 259
8.4.1 ChangeKey ... 259
8.4.2 ChangeKeyEV2 ... 263
8.4.3 InitializeKeySet ... 267
8.4.4 FinalizeKeySet ... 269
8.4.5 RollKeySet ... 271
8.4.6 GetKeySettings ... 273
8.4.7 ChangeKeySettings ... 277

Provided under NDA only. Personal copy for: COMPANY CONFIDENTIAL Sarada IT Solutions GmbH Data is not copy, share or reproduce DO 5fc07f9d-022c-4075-b00f-6ec3cae4f1

8.4.8 GetKeyVersion ... 279
8.5 Asymmetric Key Management ... 282
8.5.1 ManageKeyPair ... 282
8.5.2 ManageCARootKey ... 285
8.5.3 ExportKey ... 288
8.5.4 GetKeySettings ... 290
8.6 Application management ... 294
8.6.1 CreateApplication ... 294
8.6.2 DeleteApplication ... 297
8.6.3 CreateDelegatedApplication ... 299
8.6.4 SelectApplication ... 303
8.6.5 GetApplicationIDs ... 305
8.6.6 GetDFNames ... 307
8.6.7 GetDelegatedInfo ... 309
8.7 File Management ... 311
8.7.1 CreateStdDataFile ... 311
8.7.2 CreateBackupDataFile ... 314
8.7.3 CreateValueFile ... 316
8.7.4 CreateLinearRecordFile ... 319
8.7.5 CreateCyclicRecordFile ... 321
8.7.6 CreateTransactionMACFile ... 323
8.7.7 DeleteFile ... 326
8.7.8 GetFileIDs ... 328
8.7.9 GetISOFileIDs ... 329
8.7.10 GetFileSettings ... 331
8.7.11 GetFileCounters ... 338
8.7.12 ChangeFileSettings ... 340
8.8 Data Management ... 349
8.8.1 ReadData ... 349
8.8.2 WriteData ... 352
8.8.3 GetValue ... 355
8.8.4 Credit ... 357
8.8.5 Debit ... 359
8.8.6 LimitedCredit ... 361
8.8.7 ReadRecords ... 363
8.8.8 WriteRecord ... 366
8.8.9 UpdateRecord ... 368
8.8.10 ClearRecordFile ... 371
8.9 Transaction Management ... 373
8.9.1 CommitTransaction ... 373
8.9.2 AbortTransaction ... 375
8.9.3 CommitReaderID ... 376
8.10 Cryptographic Support ... 378
8.10.1 CryptoRequest ... 378
8.10.2 CryptoRequest_ECCSign ... 380
8.10.3 CryptoRequest_Echo ... 382
8.11 GPIO Management ... 383
8.11.1 ManageGPIO ... 383
8.11.2 ReadGPIO ... 387
8.12 ISO816-4 Support ... 389
8.12.1 ISOSelectFile ... 389
8.12.2 ISOReadBinary ... 393
8.12.3 ISOUpdateBinary ... 395
8.12.4 ISOReadRecord ... 398
8.12.5 ISOAppendRecord ... 400
8.12.6 ISOGetChallenge ... 402
8.13 Proximity Check ... 403
8.13.1 PreparePC ... 403
8.13.2 ProximityCheck ... 405
8.13.3 VerifyPC ... 407
8.14 EV Charging ... 408
8.14.1 VDE_ReadData ... 408
8.14.2 VDE_WriteData ... 410
8.14.3 VDE_ECDSASign ... 412
8.15 Originality Check ... 414
9 Limiting values ... 415
9.1 Limiting Values applicable to MOA8 ... 415
9.2 Limiting Values applicable to bumped FFC ... 415
9.3 Limiting Values applicable to WLCSP and HVQFN20 ... 415
10 Recommended operating conditions ... 417
10.1 Recommended operating conditions applicable to MOA8 ... 417
10.2 Recommended operating conditions applicable bumped FFC ... 417
10.3 Recommended operating conditions applicable to WLCSP and HVQFN20 ... 417
11 Characteristics ... 418
11.1 AC Characteristics ... 418
11.1.1 AC Characteristics applicable to WLCSP and HVQFN20 ... 419
11.2 DC Characteristics ... 419
11.2.1 DC Characteristics applicable to bumped FFC ... 419
11.2.1.1 General purpose I/O interface: GPIO1 ... 419
11.2.2 DC Characteristics applicable to WLCSP and HVQFN20 ... 420
11.2.2.1 General purpose I/O interface ... 420
11.2.2.2 I2C Interface ... 422
11.2.2.3 Power Consumption ... 423
11.2.2.4 I2C Bus Timings ... 423
11.2.3 EMC / EMI ... 423
12 Package outline ... 424
13 Abbreviations ... 426
14 References ... 431
15 Note about the source code in the document ... 433
16 Revision history ... 434
Legal information ... 436

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© 2025 NXP B.V. All rights reserved.

For more information, please visit: https://www.nxp.com

Date of release: 10 July 2025
Document identifier: MF3E(H)x3
Document number: 974412

Provided under NDA only
COMPANY CONFIDENTIAL - Personal copy for:
Quantum Sarata Quantum IT Solutions GmbH
Do not copy/share or reproduce
5fc07f9d-c22c-4057-8ee3-3ae2f1